

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)
Shahabad Daultpur, Bawana Road, Delhi- 110042

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



DATABASE MANAGEMENT SYSTEMS LAB FILE

CO-202

Submitted By:
Sandesh Shrestha
2K21/CO/417
A6 GROUP 2

Submitted To:
Dr. Indu Singh
Department of COE

INDEX: LIST OF PROGRAMS

[illegible]

[illegible]

Experiment 1

Program Objective: Introduction to SQL, Database and Database Management Systems (DBMS)

Program theory:

What is a database?

A database is information that is set up for easy access, management and updating. Computer databases typically store aggregations of data records or files that contain information, such as sales transactions, customer data, financials and product information. In simpler terms, a database is an organized collection of data, so that it can be easily accessed and managed.

The main purpose of the database is to operate a large amount of information by storing, retrieving, and managing data. Databases are used for storing, maintaining and accessing any sort of data. They collect information on people, places or things. That information is gathered in one place so that it can be observed and analyzed. Databases can be thought of as an organized collection of information.

What is database management system (DBMS)?

Database management system is a software which is used to manage the databases. For example: MySQL, Oracle, etc. are a very popular commercial database which is used in different applications.

SQL stands for Structured Query Language. It lets you access and manipulate databases. SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. It is used for storing and managing data in relational database management system (RDMS).

It is a standard language for Relational Database System. It enables a user to create, read, update and delete relational databases and tables. All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language. SQL allows users to query the database in a number of ways, using English-like statements. SQL is used to create, remove, alter the database and database objects in a database management system and to store, retrieve, update the data in a database. SQL is a standard language for creating, accessing, manipulating database management system. SQL works for all modern relational database management systems, like SQL Server, Oracle, MySQL, etc.

Applications of database management system:

There are different fields where a database management system is utilized. Following are a few applications which utilize the information base administration framework –

- **Railway Reservation System –**
In the rail route reservation framework, the information base is needed to store the record or information of ticket appointments, status about train's appearance, and

flight. Additionally, if trains get late, individuals become acquainted with it through the information base update.

- **Library Management System –**
There are lots of books in the library so; it is difficult to store the record of the relative multitude of books in a register or duplicate. Along these lines, the data set administration framework (DBMS) is utilized to keep up all the data identified with the name of the book, issue date, accessibility of the book, and its writer.
- **Banking –**
Database the executive's framework is utilized to store the exchange data of the client in the information base.
- **Education Sector –**
Presently, assessments are led online by numerous schools and colleges. They deal with all assessment information through the data set administration framework (DBMS). In spite of that understudy's enlistments subtleties, grades, courses, expense, participation, results, and so forth all the data is put away in the information base.
- **Credit card exchanges –**
The database Management framework is utilized for buying on charge cards and age of month-to-month proclamations.
- **Social Media Sites –**
We all utilization of online media sites to associate with companions and to impart our perspectives to the world. Every day, many people group pursue these online media accounts like Pinterest, Facebook, Twitter, and Google in addition to. By the utilization of the data set administration framework, all the data of clients are put away in the information base and, we become ready to interface with others.
- **Broadcast communications –**
Without DBMS any media transmission organization can't think. The Database the executive's framework is fundamental for these organizations to store the call subtleties and month to month postpaid bills in the information base.
- **Account –**
The information base administration framework is utilized for putting away data about deals, holding and acquisition of monetary instruments, for example, stocks and bonds in a data set.
- **Online Shopping –**
These days, web-based shopping has become a major pattern. Nobody needs to visit the shop and burn through their time. Everybody needs to shop through web based shopping sites, (for example, Amazon, Flipkart, Snapdeal) from home. So all the items

are sold and added uniquely with the assistance of the information base administration framework (DBMS). Receipt charges, installments, buy data these are finished with the assistance of DBMS.

- **Human Resource Management –**
Big firms or organizations have numerous specialists or representatives working under them. They store data about worker's compensation, assessment, and work with the assistance of an information base administration framework (DBMS).
- **Manufacturing –**
Manufacturing organizations make various kinds of items and deal them consistently. To keep the data about their items like bills, acquisition of the item, amount, inventory network the executives, information base administration framework (DBMS) is utilized.
- **Airline Reservation System –**
This framework is equivalent to the railroad reservation framework. This framework additionally utilizes an information base administration framework to store the records of flight takeoff, appearance, and defer status.

Advantages of Database Management System (DBMS):

Some of them are given as follows below.

- **Better Data Transferring:** Database management creates a place where users have an advantage of more and better-managed data. Thus making it possible for end-users to have a quick look and to respond fast to any changes made in their environment.
- **Better Data Security:** The more accessible and usable the database, the more it is prone to security issues. As the number of users increases, the data transferring or data sharing rate also increases thus increasing the risk of data security. It is widely used in the corporate world where companies invest money, time, and effort in large amounts to ensure data is secure and is used properly. A Database Management System (DBMS) provides a better platform for data privacy and security policies thus, helping companies to improve Data Security.
- **Better data integration:** Due to the Database Management System we have an access to well managed and synchronized form of data thus it makes data handling very easy and gives an integrated view of how a particular organization is working and also helps to keep a track of how one segment of the company affects another segment.
- **Minimized Data Inconsistency:** Data inconsistency occurs between files when different versions of the same data appear in different places. For Example, data inconsistency occurs when a student's name is saved as "John Wayne" on a main

computer of the school but on the teacher registered system same student name is “William J. Wayne”, or when the price of a product is \$86.95 in the local system of the company and its National sales office system shows the same product price as \$84.95. So if a database is properly designed then Data inconsistency can be greatly reduced hence minimizing data inconsistency.

- **Faster data Access:** The Database management system (DBMS) helps to produce quick answers to database queries thus making data access faster and more accurate. For example, to read or update the data. For example, end-users, when dealing with large amounts of sale data, will have enhanced access to the data, enabling a faster sales cycle.
- **Better decision making:** Due to DBMS now we have Better managed data and Improved data access because of which we can generate better quality information hence on this basis better decisions can be made. Better Data quality improves accuracy, validity, and time it takes to read data. DBMS does not guarantee data quality, it provides a framework to make it easy to improve data quality.
- **Increased end-user productivity:** The data which is available with the help of a combination of tools that transform data into useful information, helps end-users to make quick, informative, and better decisions that can make difference between success and failure in the global economy.
- **Simple:** Database management system (DBMS) gives a simple and clear logical view of data. Many operations like insertion, deletion, or creation of files or data are easy to implement.
- **Data abstraction:** The major purpose of a database system is to provide users with an abstract view of the data. Since many complex algorithms are used by the developers to increase the efficiency of databases that are being hidden by the users through various data abstraction levels to allow users to easily interact with the system.
- **Reduction in data Redundancy:** When working with a structured database, DBMS provides the feature to prevent the input of duplicate items in the database. for e.g. – If there are two same students in different rows, then one of the duplicate data will be deleted.

Disadvantages of using a database management system (DBMS), some of which are:

- **Complexity:** DBMSs are often complex and can be difficult to learn, configure, and manage. This can result in higher costs for training and support.
- **Cost:** Implementing and maintaining a DBMS can be expensive. There are licensing fees, hardware and software costs, and ongoing maintenance and support expenses.

- **Security risks:** DBMSs can be vulnerable to security breaches, which can lead to the loss of sensitive data. If not properly secured, hackers can exploit weaknesses in the system to access or modify data.
- **Performance:** Depending on the size and complexity of the database, DBMSs can be slow and require significant resources to operate efficiently. This can result in slower query processing and other performance issues.
- **Limited flexibility:** Some DBMSs can be restrictive in terms of the types of data they can store or the operations they can perform. This can limit the ability to customize the system to meet specific business requirements.
- **Single point of failure:** Since all data is stored in a central location, a failure of the DBMS can result in the loss of all data. This can be mitigated through backups and redundancy, but it remains a risk.
- **Vendor lock-in:** Once a business has invested in a specific DBMS, it can be difficult to switch to a different system without significant cost and disruption.

Conclusion: In conclusion, database management systems (DBMS) offer a range of advantages for businesses and organizations of all sizes, including efficient data storage, easy data access and retrieval, improved data security, and streamlined data management. However, as with any technology, DBMSs are not without their disadvantages, including complexity, cost, security risks, limited flexibility, and potential for vendor lock-in. Despite these drawbacks, the benefits of using a DBMS often outweigh the drawbacks, making them an essential tool for businesses and organizations in today's data-driven world. It is important to carefully evaluate the specific needs of your business before choosing a DBMS and to ensure that proper training, support, and security measures are in place to maximize its benefits and minimize its drawbacks.

Findings/Learning: The field of database management systems (DBMS) is constantly evolving, with new technologies and techniques being developed to improve the way data is stored, accessed, and managed.

Experiment 2

Program Objective: Introduction to various software of database management systems and database languages

Program theory:

MS SQL Server

MS SQL Server is a relational database management system (RDBMS) developed by Microsoft. This product is built for the basic function of storing retrieving data as required by other applications. It can be run either on the same computer or on another across a network. This tutorial explains some basic and advanced concepts of SQL Server such as how to create and restore data, create login and backup, assign permissions, etc. Each topic is explained using examples for easy understanding. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet). Microsoft markets at least a dozen different editions of Microsoft SQL Server, aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users.



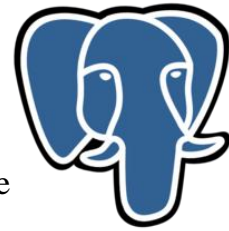
Oracle

Oracle database is a relational database management system (RDBMS) from Oracle Corporation. This article will explain a complete overview of the Oracle database, features, history, and editions. Before discussing the oracle, we will first need to know about the database. Oracle database is a relational database management system. It is also called OracleDB, or simply Oracle. It is produced and marketed by Oracle Corporation. It was created in 1977 by Lawrence Ellison and other engineers. It is one of the most popular relational database engines in the IT market for storing, organizing, and retrieving data. Oracle database was the first DB that designed for enterprise grid computing and data warehousing. Enterprise grid computing provides the most flexible and cost-effective way to manage information and applications. It uses SQL queries as a language for interacting with the database.



PostgreSQL

PostgreSQL also known as Postgres, was developed by Michael Stonebraker of the University of California, Berkley. It started as the Ingres Project and later evolved into PostgreSQL as we know today. In the year 1982, Michael Stonebraker started a post-Ingres project to address the problems with contemporary database systems. He was awarded the Turing Award in the year 2014 for the projects and techniques pioneered in them. PostgreSQL is one of the most advanced general-purpose object-relational database management system and is open-source. Being an open-source software, its source code is available under PostgreSQL license, a liberal open-source license. Anyone with the right skills is free to use, modify, and distribute PostgreSQL in any form. As it is highly stable, very low effort is required to maintain this DBMS.



PostgreSQL

MySQL

MySQL is a database management system. A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications. The MySQL Database Software is a client/server system that consists of a multithreaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs). MySQL software is Open Source. Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything.



MySQL®

Oracle 10G Express Edition:

Oracle Database 10g Express Edition (Oracle Database XE) is a free, downloadable version of the world's most capable relational database. Oracle Database XE is easy to install and easy to manage. With Oracle Database XE, you use the Database Home Page, an intuitive browser-based interface, to administer the database; create tables, views, and other schema objects; import, export, and view table data; run queries and SQL scripts; and generate reports. Oracle Database XE includes Oracle HTML DB 2.1, a declarative,

graphical development environment for creating database-centric Web applications. In addition to HTML DB 2.1, you can use popular Oracle and third-party languages and tools to develop your Oracle Database XE applications.



Conclusion: In conclusion, the study of database management systems and database languages is essential for anyone who wants to work with data. These tools are used to store, manipulate, and analyze data, which is essential for decision-making and business operations.

Findings/Learning: Findings/Learning in this area could include advanced topics such as database normalization, indexing, stored procedures, and triggers. Additionally, learning about other database management systems such as NoSQL databases, object-oriented databases, and graph databases could provide valuable insights for specific use cases. Finally, keeping up with the latest developments in this field, such as new tools, languages, and technologies, could help professionals stay relevant and effective in their roles.

Experiment 3

Program Objective: Introduction to Entity Relationship diagram (ER Diagram)

Program theory:

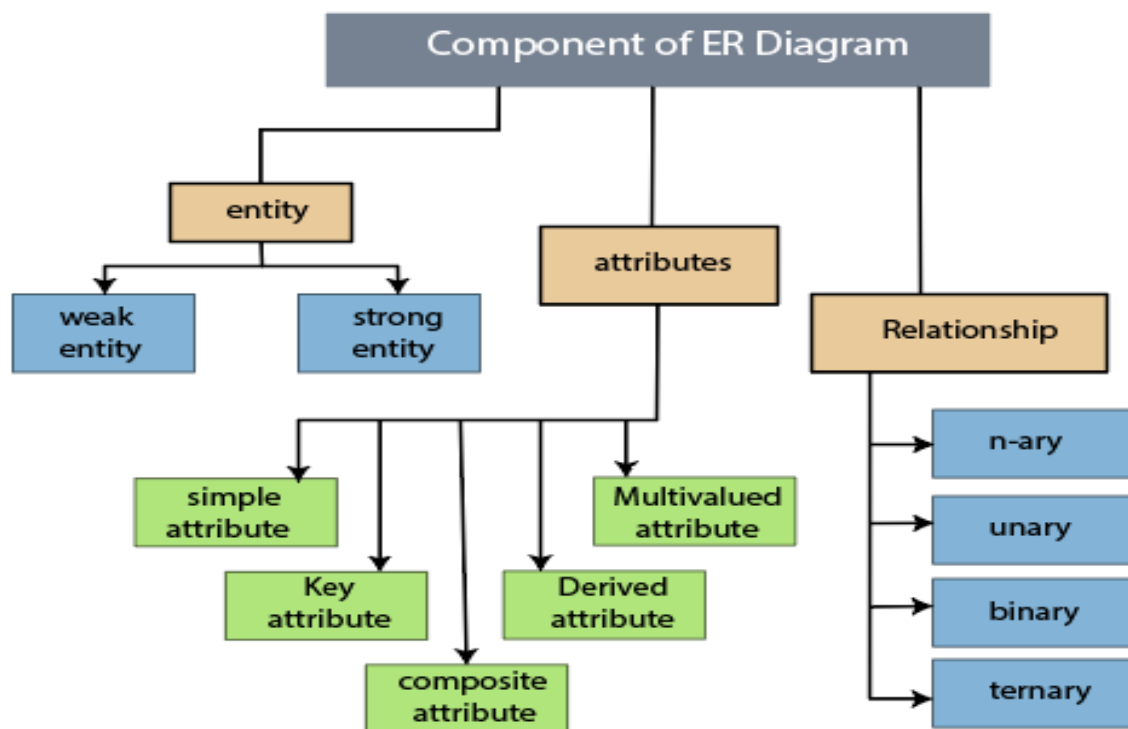
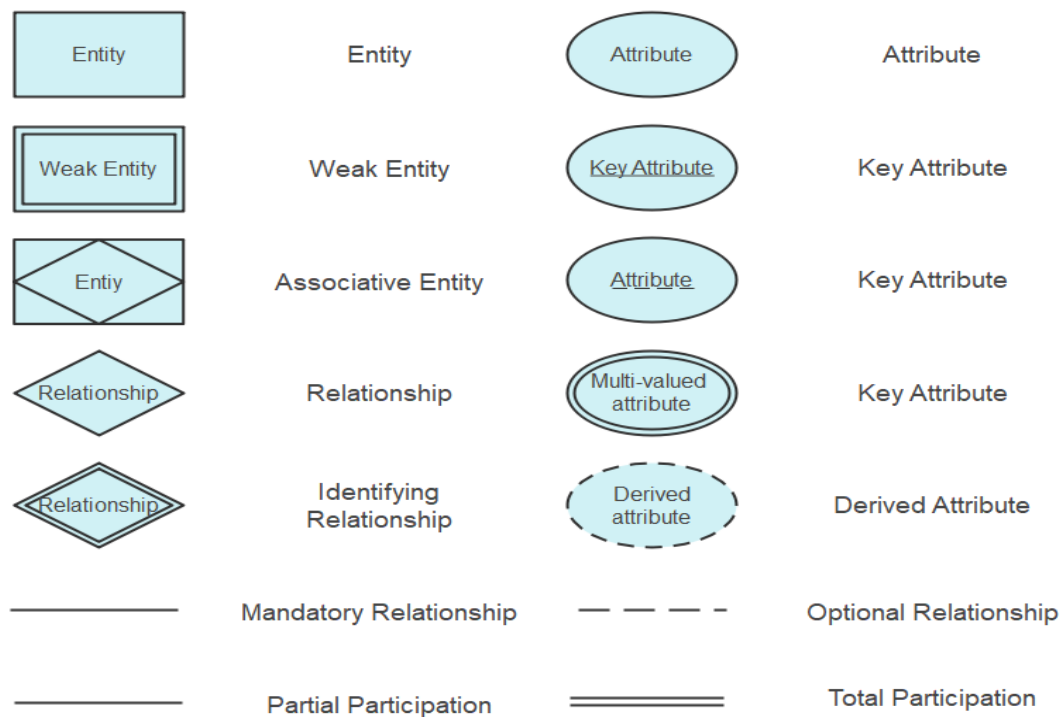
ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data. In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram. An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research.

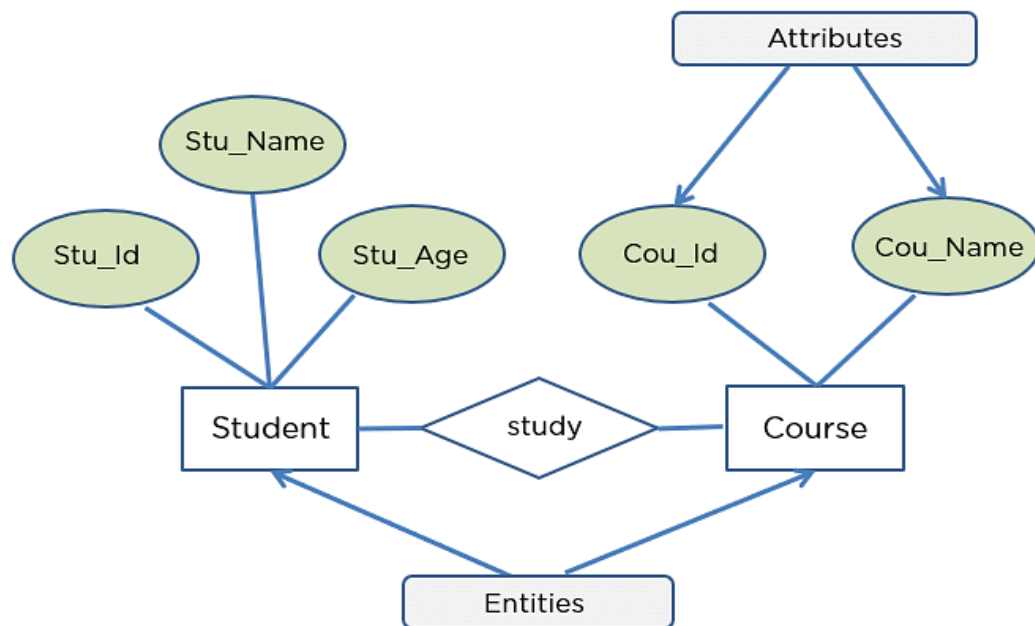
- An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
- The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.
- A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

The advantages of using an ER diagram include:

- **Clarity:** ER diagrams provide a clear and concise way to represent the relationships between entities. This makes it easier for stakeholders to understand the relationships and how they fit into the overall system.
- **Communication:** ER diagrams can be used to communicate complex ideas and relationships to both technical and non-technical stakeholders.
- **Flexibility:** ER diagrams can be easily modified to reflect changes in the system or to add new entities and relationships.
- **Database Design:** ER diagrams can be used as a blueprint for designing a database schema. By visualizing the relationships between entities, designers can create a more efficient and effective database.
- **Data Integrity:** ER diagrams help ensure data integrity by identifying relationships between entities that must be maintained for the system to work correctly.
- **Integration:** ER diagrams can be used to integrate multiple systems by identifying the relationships between entities in different systems and ensuring that they are compatible.

- Analysis: ER diagrams can be used to analyze the system and identify potential areas for improvement or optimization.





ER DIAGRAM EXAMPLE

Conclusion: In conclusion, Entity Relationship diagrams (ER diagrams) are a powerful tool for visualizing and designing database structures. By representing entities and their relationships, ER diagrams help developers understand the database's overall structure and identify potential design issues before implementation.

Findings/Learning: Further learning in ER diagrams may involve gaining a deeper understanding of the different types of relationships, such as one-to-one, one-to-many, and many-to-many, and how to properly represent them in a diagram.

Experiment 4

Program Objective: Different types of constraints in SQL

Program theory:

Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

Syntax:

```
CREATE TABLE table_name (  
column1 datatype constraint,  
column2 datatype constraint,  
column3 datatype constraint,  
....  
);
```

Constraints in SQL can be categorized into two types:

- a) **Column Level Constraint:** Column Level Constraint is used to apply a constraint on a single column.
- b) **Table Level Constraint:** Table Level Constraint is used to apply a constraint on multiple columns.

SQL Constraints

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**

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.

Syntax:

```
CREATE TABLE TableName (  
ColumnName1 datatype NOT NULL,  
ColumnName2 datatype,  
ColumnNameN datatype);
```

- **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.

Syntax:

```
CREATE TABLE TableName (  
  ColumnName1 datatype UNIQUE,  
  ColumnName2 datatype, ...,  
  ColumnNameN datatype);
```

- **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.

Syntax:

```
CREATE TABLE TableName (ColumnName1 datatype PRIMARY KEY,  
  ColumnName2 datatype, ..., ColumnNameN datatype);
```

- **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.

Syntax:

```
CREATE TABLE tablename (ColumnName1 Datatype(SIZE) PRIMARY KEY,  
  ColumnNameN Datatype(SIZE), FOREIGN KEY( ColumnName ) REFERENCES  
  PARENT_TABLE_NAME(Primary_Key_ColumnName));
```

- **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. For example: if we have an age column in a table, then the user will insert any value of his choice. The user will also enter even a negative value or any other invalid value. But if the user has applied check constraint on the age column with the condition age greater than 18. Then in such cases, even if a user tries to insert an invalid value such as zero or any other value less than 18, then the age column will not accept that value and will not allow the user to insert it due to the application of check constraint on the age column.

Syntax:

```
CREATE TABLE TableName (ColumnName1 datatype CHECK (ColumnName1  
  Condition), ColumnName2 datatype, ..., ColumnNameN datatype);
```


- **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.

Syntax:

```
CREATE TABLE TableName (ColumnName1 datatype DEFAULT Value,  
ColumnName2 datatype,..., ColumnNameN datatype);
```

- **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.

Syntax:

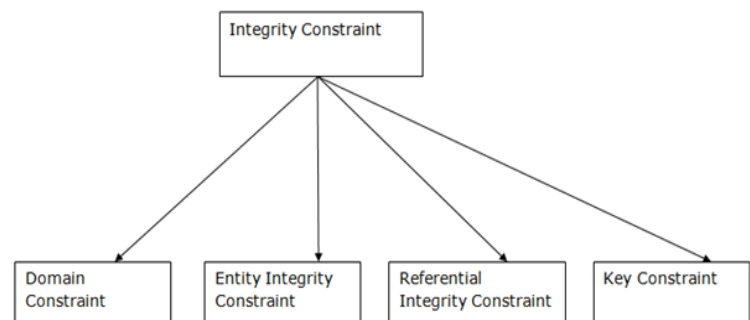
```
CREATE INDEX IndexName ON TableName (ColumnName 1);
```

Integrity Constraints

Integrity constraints are a set of rules. It is used to maintain the quality of information.

Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.

Thus, integrity constraint is used to guard against accidental damage to the database.



Domain constraints

Domain constraints can be defined as the definition of a valid set of values for an attribute. The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Entity integrity constraints

The entity integrity constraint states that primary key value can't be null.

This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.

A table can contain a null value other than the primary key field.

Referential Integrity Constraints

A referential integrity constraint is specified between two tables.

In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Key constraints

Keys are the entity set that is used to identify an entity within its entity set uniquely.

An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Conclusion: In conclusion, understanding different types of constraints in SQL is essential for designing and maintaining a database. Constraints ensure data integrity and consistency by restricting the type of data that can be entered into a database. The primary key constraint ensures that each record in a table is uniquely identified, while the foreign key constraint maintains the relationship between tables. The unique constraint ensures that each value in a column is unique, while the check constraint ensures that the data entered meets certain criteria. Finally, the not null constraint ensures that a column cannot have a null value.

Findings/Learning: For Findings/Learning, one can dive deeper into the different types of constraints and how they can be implemented in SQL to design more complex databases. Additionally, understanding the trade-offs and considerations when selecting different types of constraints for a database can help in designing more efficient and effective systems.

Experiment 5

Program Objective: Case Study Hospital Management Database System

Program theory:

Hospital management system is a computer system that helps manage the information related to health care and aids in the job completion of health care providers effectively. Hospital Management System brings together all the information and processes of a hospital, in a single platform. It presents you with a unified 360-degree view for managing patients, doctors, inventory, appointments, billing information, finances and much more.

The Hospital Management System consists of the following components/table: -

1. PRESCRIPTION REPORT TABLE

It will manage and maintains all the information about all the prescriptions made by doctor like dept_name, cont_no, hod_id, etc.

2. DOCTOR TABLE

It will maintain information about the doctors like name, id, cont_no, address, qualifications, specifications, etc.

3. PATIENT TABLE

It will maintain all the information about all the patients like their name, age, address, sex, date of birth, admit date, disease they are infected with, total tests done, contact no, etc.

4. MEDICINE TABLE

It will maintain the information about all the medicines issued to patients in the prescription.

5. APPOINTMENTS TABLE

It will store the information regarding the tests that had been done by each patient like test name, patient name, test charge, etc.

6. BILLS TABLE

It will maintain the information about all the charges that the patient is required to pay like bed charge, tests charges, etc.

7. HOSPITAL TABLE

It will maintain the information about all the hospitals, their branches and everything related to that.

ASSUMPTIONS:

1. A Department can have many doctors but a doctor will work in only one Department.
2. A Department can have many nurses but a nurse will work in only one department.
3. A department may have multiple contact numbers.
4. Doctors and nurses may visit the patients admitted in the words of different department.

Identifications of entities and their respective attributes:

1. **PRESCRIPTION REPORT:** reportID, reportdescription, doctor
2. **DOCTOR:** docId, docName, gender, contactNo, specifications, qualifications, address
3. **MEDICINE:** medicineid, medicinename, price, reportid
4. **PATIENTS:** patientId, patientName, gender, admitDate, patientAge, contactNo, qualifications, bedNo, address
5. **BILLS:** BillID, PatientID, total cost, billdate, duedate
6. **APPOINTMENTS:** AppointmentID, appointmenttime, appointmentdate, patient i

Conclusion:

In conclusion, the implementation of a hospital database management system leads to significant improvements in the hospital's operations. The system helps streamline patient care, reduce errors, improve efficiency, and comply with regulatory requirements. The hospital management pleased with the system's performance and continued to work with the development team to ensure it remained up to date with changing requirements.

Findings/Learning:

Firstly, the system improves the accuracy of patient records and reduced errors caused by manual data entry. Secondly, doctors and nurses will be able to access patient records quickly, leading to faster diagnoses and treatment. Thirdly, the system helps the hospital management monitor the performance of staff and operations, leading to improved efficiency and cost savings. Fourthly, the system helps the hospital comply with regulatory requirements for patient data management. Lastly, the system improves patient satisfaction by reducing waiting times and improving the overall quality of care.

Experiment 6

Program Objective: Creating ER Diagram for the Hospital Management System

Program theory:

ER diagrams mainly comprise of – Entity and its attributes, Relationship, which is association among entities.

There are several tables that may be required for a hospital management system, depending on the specific needs and requirements of the organization. Here are some common examples:

Patient table: This table stores information about each patient, including their name, address, phone number, date of birth, medical history, and any allergies or medications they are taking.

Doctor table: This table stores information about each doctor or healthcare provider, including their name, specialty, contact information, and schedule.

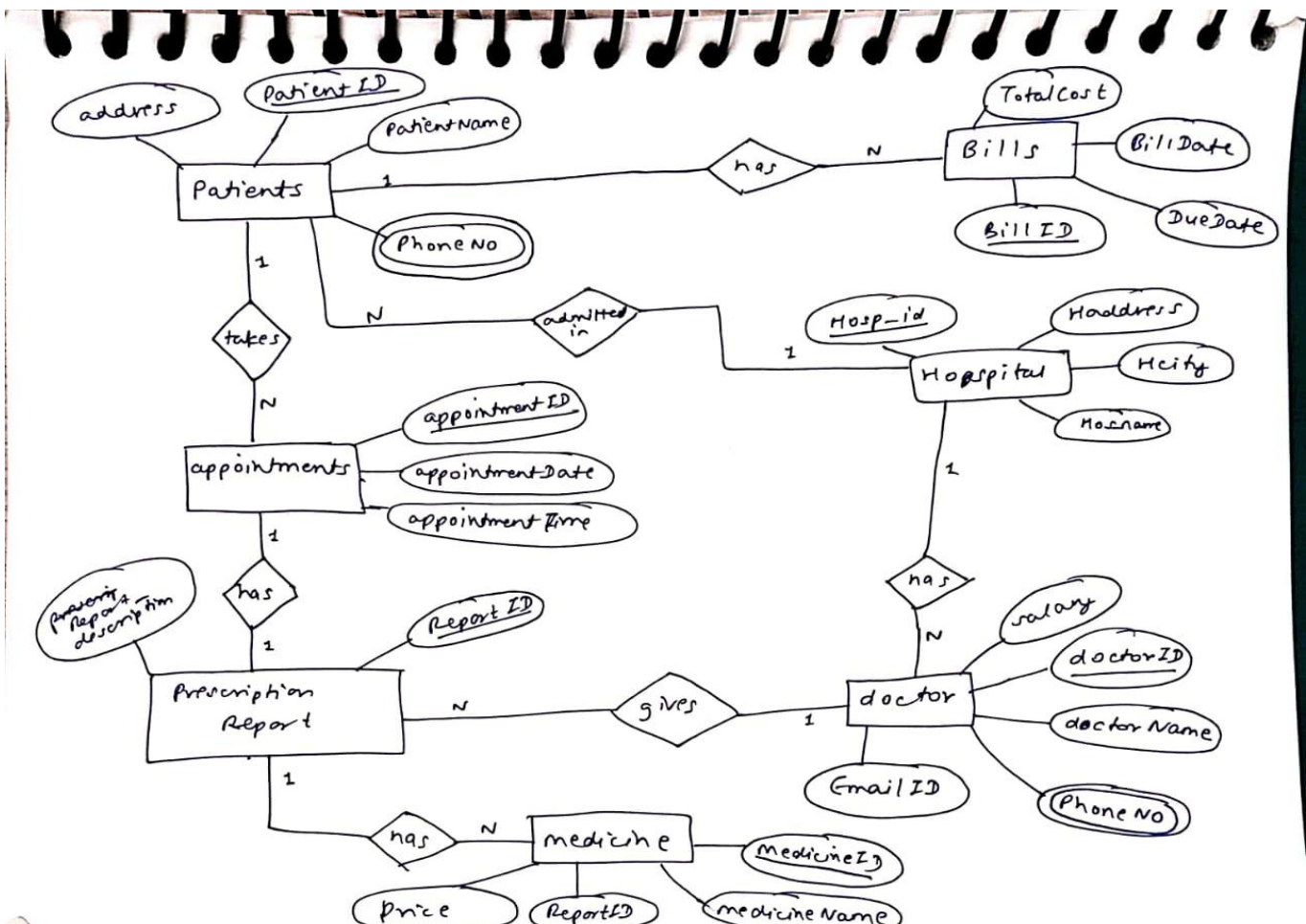


Figure ER DIAGRAM FOR HOSPITAL MANAGEMENT SYSTEM

Appointment table: This table tracks appointments between patients and doctors, including the date and time of the appointment, the doctor and patient involved, and any notes or comments.

Medical record table: This table stores the medical history of each patient, including diagnoses, treatments, and medications prescribed.

Billing table: This table stores information about each patient's bill, including the services rendered, insurance information, and payment status.

Staff table: This table stores information about hospital staff members, including their name, job title, contact information, and schedule.

Department table: This table stores information about hospital departments, including their name, location, and manager.

These are just a few examples of the types of tables that might be included in a hospital management system. The specific tables required may vary depending on the needs and size of the hospital, as well as any regulatory or legal requirements.

To convert an **ER diagram into a set of relational tables**, you need to follow these steps:

- Identify all the entities in the ER diagram and create a table for each entity.
- Identify all the attributes for each entity and add them as columns to the corresponding table.
- Identify all the relationships between the entities and create tables for the relationship entities (if needed).
- For each one-to-one relationship, add the primary key of one entity as a foreign key in the table of the other entity.
- For each one-to-many relationship, add the primary key of the "one" entity as a foreign key in the table of the "many" entities.
- For each many-to-many relationship, create a new table that includes the primary keys of both entities involved in the relationship.

Relational table

Patient's Table

<u>Patient-ID</u>	Patient Name	Phone no.	address	Hosp-id ^{FK}
-------------------	--------------	-----------	---------	-----------------------

Doctor Table

<u>Doctor ID</u>	Doctor Name	Phone No	Email ID	Hosp-id ^{FK}
------------------	-------------	----------	----------	-----------------------

Prescription Report Table

<u>Report ID</u>	Report Desc	Doctor ID	salary
------------------	-------------	-----------	--------

Medicine Table

<u>Medicine ID</u>	Medicine Name	Price	Report ID ^{FK}
--------------------	---------------	-------	-------------------------

Appointments Table

<u>Appointment ID</u>	Appointment Date	Appointment Time
Patient ID ^{FK}	Report ID ^{FK}	

Bills Table

<u>Bill ID</u>	Patient ID ^{FK}	Total Cost	Bill Date	Due Date
----------------	--------------------------	------------	-----------	----------

Hospital Table

<u>Hosp-id</u>	Haddress	Hcity	Hosprname
----------------	----------	-------	-----------

Conclusion : In general, an ER diagram is a visual representation of the relationships between entities in a database. It is a useful tool for designing and understanding complex data models, and can be used to ensure that a database is well-structured, normalized, and optimized for performance.

Findings/Learning: ER diagram is an important part of database design and can help to ensure that a database is well-organized, efficient, and effective. By using an ER diagram, designers can gain a better understanding of the relationships between entities and can create a database that is optimized for the needs of its users.

Experiment 7

Program Objective: Implementing DDL and DML commands

Program theory:

1) DDL : Data Definition Language.

The DDL Commands in Structured Query Language are used to create and modify the schema of the database and its objects. The syntax of DDL commands is predefined for describing the data. The commands of Data Definition Language deal with how the data should exist in the database.

Following are the five DDL commands in SQL:

- CREATE Command
- DROP Command
- ALTER Command
- TRUNCATE Command
- RENAME Command

a) CREATE Command

CREATE is a DDL command used to create databases, tables, triggers and other database objects.

Syntax to create a new table:

```
CREATE TABLE table_name (
column_Name1 data_type ( size of the column ) ,
column_Name2 data_type ( size of the column ) ,
column_Name3 data_type ( size of the column ) ,
...
column_NameN data_type ( size of the column ) ;
```

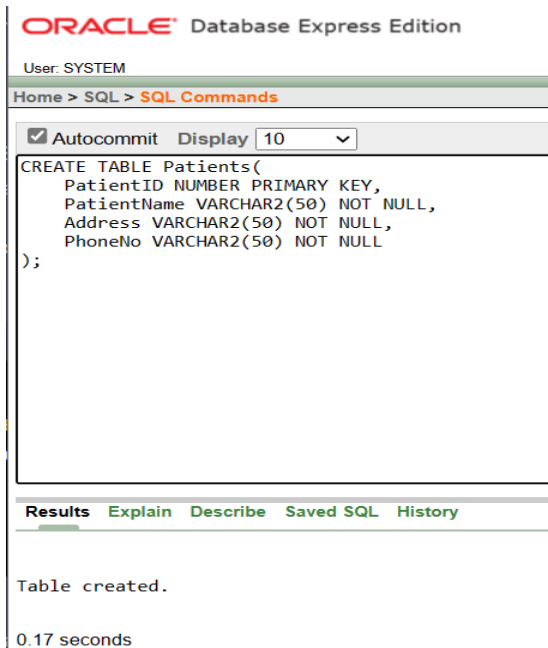


Figure 1 creating patients table

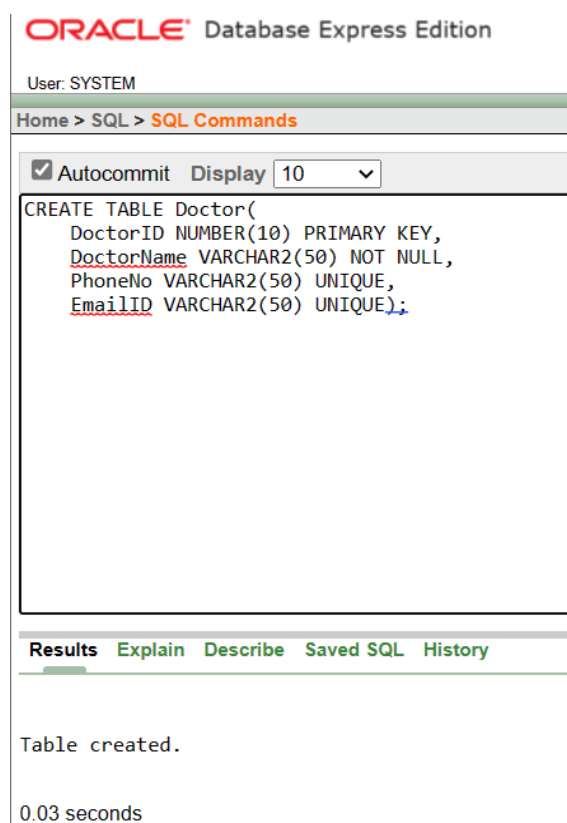


Figure 2 creating doctor table

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10

```
CREATE TABLE PrescriptionReport(  
  ReportID NUMBER(10) PRIMARY KEY,  
  ReportDescription VARCHAR2(255) DEFAULT 'General Checkup',  
  DoctorID NUMBER(10),  
  FOREIGN KEY(DoctorID) REFERENCES Doctor(DoctorID));
```

Results Explain Describe Saved SQL History

Table created.

0.00 seconds

Language: en-us Copyright © 19

Figure 3 creating prescription report table

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10

```
CREATE TABLE Medicine(  
  MedicineID NUMBER(10) PRIMARY KEY,  
  MedicineName VARCHAR2(255) NOT NULL,  
  Price VARCHAR2(255) NOT NULL,  
  ReportID NUMBER(10),  
  FOREIGN KEY(ReportID) REFERENCES PrescriptionReport(ReportID)  
);
```

Results Explain Describe Saved SQL History

Table created.

0.02 seconds

Figure 4 creating medicine table

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10

```
CREATE TABLE Appointments(  
  AppointmentID NUMBER(10) PRIMARY KEY,  
  AppointmentDate DATE NOT NULL,  
  AppointmentTime VARCHAR2(30) NOT NULL,  
  PatientID NUMBER(10),  
  ReportID NUMBER(10),  
  FOREIGN KEY(PatientID) REFERENCES Patients(PatientID),  
  FOREIGN KEY(ReportID) REFERENCES PrescriptionReport(ReportID));
```

Results Explain Describe Saved SQL History

Table created.

0.03 seconds

Figure 5 creating appointment table

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10 Save Run

describe Patients;

Results Explain Describe Saved SQL History

Object Type TABLE Object PATIENTS

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PATIENTS	PATIENTID	Number	-	-	-	1	-	-	-
	PATIENTNAME	Varchar2	50	-	-	-	-	-	-
	ADDRESS	Varchar2	50	-	-	-	-	-	-
	PHONENO	Varchar2	50	-	-	-	-	-	-

1 - 4

Figure 6 describe patients table

☒ Autocommit Display 10 Save Run

describe Medicine;

Results Explain Describe Saved SQL History

Object Type TABLE Object MEDICINE

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MEDICINE	MEDICINEID	Number	-	10	0	1	-	-	-
	MEDICINENAME	Varchar2	255	-	-	-	-	-	-
	PRICE	Varchar2	255	-	-	-	-	-	-
	REPORTID	Number	-	10	0	-	✓	-	-

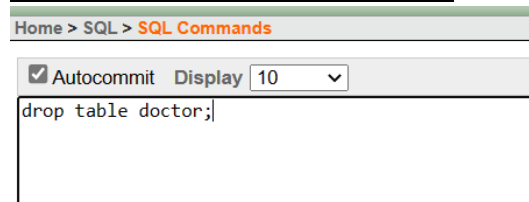
1 - 4

Figure 7 describe medicine table

b) DROP Command

DROP is a DDL command used to delete/remove the database objects from the SQL database. We can easily remove the entire table, view, or index from the database using this DDL command.

Syntax to remove a database: DROP DATABASE Database_Name;



Home > SQL > SQL Commands

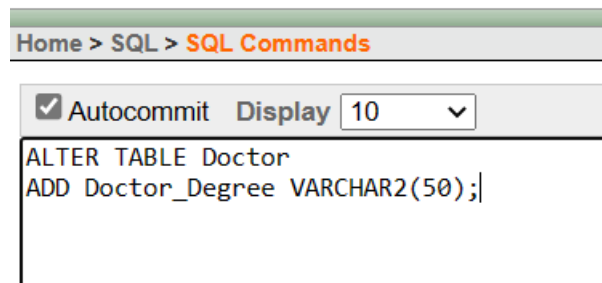
☒ Autocommit Display 10 ▼

```
drop table doctor;
```

c) ALTER Command

ALTER is a DDL command which changes or modifies the existing structure of the database, and it also changes the schema of database objects.

Syntax to add a new field in the table: ALTER TABLE name of table ADD column name column_definition;



Home > SQL > SQL Commands

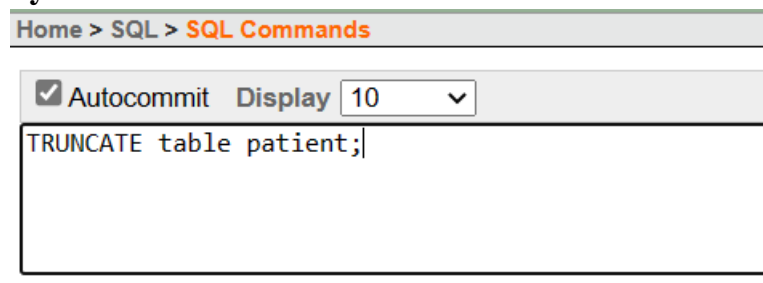
☒ Autocommit Display 10 ▼

```
ALTER TABLE Doctor  
ADD Doctor_Degree VARCHAR2(50);
```

d) TRUNCATE Command

TRUNCATE is another DDL command which deletes or removes all the records from the table. This command also removes the space allocated for storing the table records

Syntax of TRUNCATE command: TRUNCATE TABLE Table_Name;



Home > SQL > SQL Commands

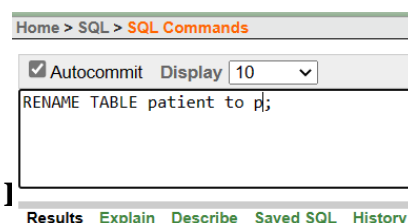
☒ Autocommit Display 10 ▼

```
TRUNCATE table patient;
```

e) RENAME Command

RENAME is a DDL command which is used to change the name of the database table.

Syntax of RENAME command : RENAME TABLE Old_Table_Name TO New_Table_Name;



Home > SQL > SQL Commands

☒ Autocommit Display 10 ▼

```
RENAME TABLE patient to p;
```

Results Explain Describe Saved SQL History

2) DML Commands in SQL

DML is an abbreviation of Data Manipulation Language

The DML commands in Structured Query Language change the data present in the SQL database. We can easily access, store, modify, update and delete the existing records from the database using DML commands.

Following are the four main DML commands in SQL:

- SELECT Command
- INSERT Command
- UPDATE Command
- DELETE Command

a) SELECT DML Command

SELECT is the most important data manipulation command in Structured Query Language. The SELECT command shows the records of the specified table. It also shows the particular record of a particular column by using the WHERE clause.

Syntax of SELECT DML command

```
SELECT column_Name_1, column_Name_2, ....., column_Name_N FROM  
Name_of_table;
```

```
SELECT * FROM table_name;
```

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10 ▼

```
select* from patients;
```

Results Explain Describe Saved SQL History

PATIENTID	PATIENTNAME	ADDRESS	PHONENO
1	Liam Johnson	123 Main St, Anytown	123-456-7890
2	Ava Smith	456 Oak Ave, Another Town	555-555-1234
3	Noah Davis	789 Pine Rd, Big City	111-222-3333
4	Sophia Rodriguez	321 Maple Dr, Smallville	444-555-6666
5	William Brown	987 Cherry Ln, Metropolis	777-888-9999
6	Isabella Wilson	654 Elm St, Gotham	333-444-5555
7	James Lee	246 5th Ave, Springfield	888-999-0000
8	Olivia Perez	135 Broadway, Capital City	222-333-4444
9	Oliver Martin	864 High St, Atlantis	111-222-3333
10	Lalu Yadav	Bihar	987-634-259

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds

[CSV Export](#)

Home > SQL > SQL Commands

☒ Autocommit Display 10 ▼

```
select* from PrescriptionReport;
```

Results Explain Describe Saved SQL History

REPORTID	REPORTDESCRIPTION	DOCTORID
1	Leg Injury	1
2	Heart Attack	4
3	Broken Teeth	5
4	Typhoid	2
5	Fever and Commoncold	3
6	Hand Injury	1

6 rows returned in 0.02 seconds

[CSV Export](#)

☒ Autocommit Display 10

select* from appointments;

Results Explain Describe Saved SQL History

APPOINTMENTID	APPOINTMENTDATE	APPOINTMENTTIME	PATIENTID	REPORTID
1	11-APR-22	10AM	1	1
2	13-APR-22	10AM	2	2
3	09-APR-22	10AM	3	3
4	08-APR-22	10AM	4	4
5	13-APR-22	10AM	5	5
6	20-APR-22	10AM	1	6

6 rows returned in 0.00 seconds [CSV Export](#)

Home > SQL > SQL Commands

☒ Autocommit Display 10

select* from medicine;

Results Explain Describe Saved SQL History

MEDICINEID	MEDICINENAME	PRICE	REPORTID
1	Aspirin	Rs.200	1
2	Vasoter	Rs.250	2
3	Avandia	Rs.300	2
4	OraGel	Rs.179	3
5	Ceftriaxone	Rs.200	4
6	Vivotif	Rs.250	4
7	Paracetamol	Rs.60	5
8	Moov	Rs.70	6

8 rows returned in 0.00 seconds [CSV Export](#)

☒ Autocommit Display 10

select* from doctor;

Results Explain Describe Saved SQL History

DOCTORID	DOCTORNAME	PHONENO	EMAILID
1	Dr. Jackson	555-123-4567	jackson@hospital.com
2	Dr. Smith	555-234-5678	smith@hospital.com
3	Dr. Patel	555-345-6789	patel@hospital.com
4	Dr. Chen	555-456-7890	chen@hospital.com
5	Dr. Kim	555-567-8901	kim@hospital.com

5 rows returned in 0.00 seconds [CSV Export](#)

☒ Autocommit Display 10

select* from bills;

Results Explain Describe Saved SQL History

BILLID	PATIENTID	TOTALCOST	BILLDATE	DUE DATE
1	2	100	01-MAR-23	31-MAR-23
2	3	250	01-MAR-23	31-MAR-23
3	4	75.5	01-MAR-23	31-MAR-23
4	5	500	01-MAR-23	31-MAR-23
5	1	1000	01-MAR-23	31-MAR-23
6	6	200	01-MAR-23	31-MAR-23

6 rows returned in 0.00 seconds [CSV Export](#)

b) INSERT DML Command

INSERT is another most important data manipulation command in Structured Query Language, which allows users to insert data in database tables.

Syntax of INSERT Command: INSERT INTO TABLE_NAME (column_Name1 , column_Name2 , column_Name3 , column_NameN) VALUES (value_1, value_2, value_3, value_N) ;

Home > SQL > SQL Commands

☒ Autocommit Display 10

```
INSERT ALL
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (1, 'Liam Johnson', '123 Main St, Anytown', '123-456-7890')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (2, 'Ava Smith', '456 Oak Ave, Another Town', '555-555-1234')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (3, 'Noah Davis', '789 Pine Rd, Big City', '111-222-3333')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (4, 'Sophia Rodriguez', '321 Maple Dr, Smallville', '444-555-6666')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (5, 'William Brown', '987 Cherry Ln, Metropolis', '777-888-9999')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (6, 'Isabella Wilson', '654 Elm St, Gotham', '333-444-5555')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (7, 'James Lee', '246 5th Ave, Springfield', '888-999-0000')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (8, 'Olivia Perez', '135 Broadway, Capital City', '222-333-4444')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (9, 'Oliver Martin', '864 High St, Atlantis', '111-222-3333')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (10, 'Lalu Yadav', 'Bihar', '987-634-259')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (11, 'Rahul Sharma', 'Delhi', '767-473-5890')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (12, 'Aman Yadav', 'Delhi', '858-7848-789')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (13, 'Sameer Saxena', 'Mumbai', '838-146-7671')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (14, 'Akash Chopra', 'Goa', '734-932-1587')
INTO Patients(PatientID, PatientName, Address, PhoneNo) VALUES (15, 'Emma Thompson', '579 Low St, Neverland', '555-444-3333')
SELECT 1 FROM dual;
```

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10

```
INSERT ALL
  INTO Doctor(DoctorID, DoctorName, PhoneNo, EmailID) VALUES(1, 'Dr. Jackson', '555-123-4567', 'jackson@hospital.com')
  INTO Doctor(DoctorID, DoctorName, PhoneNo, EmailID) VALUES(2, 'Dr. Smith', '555-234-5678', 'smith@hospital.com')
  INTO Doctor(DoctorID, DoctorName, PhoneNo, EmailID) VALUES(3, 'Dr. Patel', '555-345-6789', 'patel@hospital.com')
  INTO Doctor(DoctorID, DoctorName, PhoneNo, EmailID) VALUES(4, 'Dr. Chen', '555-456-7890', 'chen@hospital.com')
  INTO Doctor(DoctorID, DoctorName, PhoneNo, EmailID) VALUES(5, 'Dr. Kim', '555-567-8901', 'kim@hospital.com')
SELECT * FROM dual;
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

5 row(s) inserted.

0.00 seconds

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10

```
INSERT ALL
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (1, 2, 100.00, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (2, 3, 250.00, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (3, 4, 75.50, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (4, 5, 500.00, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (5, 1, 1000.00, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (6, 6, 200.00, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
SELECT * FROM dual;
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

6 row(s) inserted.

0.00 seconds

Home > SQL > SQL Commands

☒ Autocommit Display 10

```
INSERT ALL
  INTO PrescriptionReport(ReportID, ReportDescription, DoctorID) VALUES(1, 'Leg Injury', 1)
  INTO PrescriptionReport(ReportID, ReportDescription, DoctorID) VALUES(2, 'Heart Attack', 4)
  INTO PrescriptionReport(ReportID, ReportDescription, DoctorID) VALUES(3, 'Broken Teeth', 5)
  INTO PrescriptionReport(ReportID, ReportDescription, DoctorID) VALUES(4, 'Typhoid', 2)
  INTO PrescriptionReport(ReportID, ReportDescription, DoctorID) VALUES(5, 'Fever and Commoncold', 3)
  INTO PrescriptionReport(ReportID, ReportDescription, DoctorID) VALUES(6, 'Hand Injury', 1)
SELECT * FROM dual;
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

6 row(s) inserted.

0.00 seconds

☒ Autocommit Display 10

```
INSERT ALL
  INTO Medicine(MedicineID, MedicineName, Price, ReportID) VALUES(1, 'Aspirin', 'Rs.200', 1)
  INTO Medicine(MedicineID, MedicineName, Price, ReportID) VALUES(2, 'Vasoter', 'Rs.250', 2)
  INTO Medicine(MedicineID, MedicineName, Price, ReportID) VALUES(3, 'Avandia', 'Rs.300', 2)
  INTO Medicine(MedicineID, MedicineName, Price, ReportID) VALUES(4, 'OraGel', 'Rs.179', 3)
  INTO Medicine(MedicineID, MedicineName, Price, ReportID) VALUES(5, 'Ceftriaxone', 'Rs.200', 4)
  INTO Medicine(MedicineID, MedicineName, Price, ReportID) VALUES(6, 'Vivotif', 'Rs.250', 4)
  INTO Medicine(MedicineID, MedicineName, Price, ReportID) VALUES(7, 'Paracetamol', 'Rs.60', 5)
  INTO Medicine(MedicineID, MedicineName, Price, ReportID) VALUES(8, 'Moov', 'Rs.70', 6)
SELECT * FROM dual;
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

Home > SQL > [SQL Commands](#)

☒ Autocommit Display 10

```
INSERT ALL
  INTO Appointments(AppointmentID, AppointmentDate, AppointmentTime, PatientID, ReportID)
  VALUES(1, TO_DATE('2022-04-11', 'YYYY-MM-DD'), '10AM', 1, 1)
  INTO Appointments(AppointmentID, AppointmentDate, AppointmentTime, PatientID, ReportID)
  VALUES(2, TO_DATE('2022-04-13', 'YYYY-MM-DD'), '10AM', 2, 2)
  INTO Appointments(AppointmentID, AppointmentDate, AppointmentTime, PatientID, ReportID)
  VALUES(3, TO_DATE('2022-04-09', 'YYYY-MM-DD'), '10AM', 3, 3)
  INTO Appointments(AppointmentID, AppointmentDate, AppointmentTime, PatientID, ReportID)
  VALUES(4, TO_DATE('2022-04-08', 'YYYY-MM-DD'), '10AM', 4, 4)
  INTO Appointments(AppointmentID, AppointmentDate, AppointmentTime, PatientID, ReportID)
  VALUES(5, TO_DATE('2022-04-13', 'YYYY-MM-DD'), '10AM', 5, 5)
  INTO Appointments(AppointmentID, AppointmentDate, AppointmentTime, PatientID, ReportID)
  VALUES(6, TO_DATE('2022-04-20', 'YYYY-MM-DD'), '10AM', 1, 6)
SELECT 1 FROM dual;
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

6 row(s) inserted.

User: SYSTEM

Home > SQL > [SQL Commands](#)

☒ Autocommit Display 10

```
INSERT ALL
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (1, 2, 100.00, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (2, 3, 250.00, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (3, 4, 75.50, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (4, 5, 500.00, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (5, 1, 1000.00, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
  INTO Bills (BillID, PatientID, TotalCost, BillDate, DueDate) VALUES (6, 6, 200.00, TO_DATE('2023-03-01', 'YYYY-MM-DD'), TO_DATE('2023-03-31', 'YYYY-MM-DD'))
SELECT * FROM dual;
```

[Results](#) [Explain](#) [Describe](#) [Saved SQL](#) [History](#)

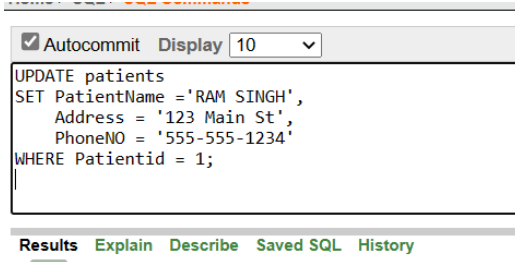
6 row(s) inserted.

0.00 seconds

c) UPDATE DML Command

UPDATE is another most important data manipulation command in Structured Query Language, which allows users to update or modify the existing data in database tables.

Syntax of UPDATE Command: UPDATE Table_name SET [column_name1= value_1,, column_nameN = value_N] WHERE CONDITION;



```
Autocommit Display 10
UPDATE patients
SET PatientName ='RAM SINGH',
    Address = '123 Main St',
    PhoneNO = '555-555-1234'
WHERE Patientid = 1;
```

Results Explain Describe Saved SQL History

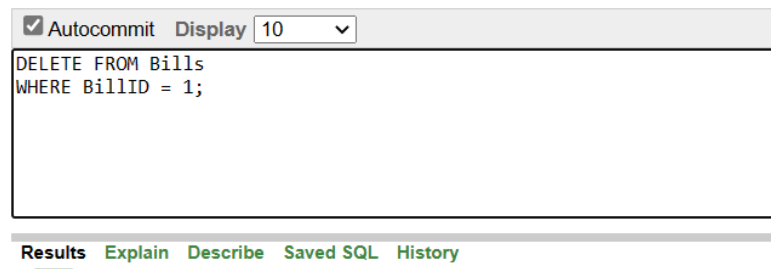
1 row(s) updated.

0.01 seconds

d) DELETE DML Command

DELETE is a DML command which allows SQL users to remove single or multiple existing records from the database tables. This command of Data Manipulation Language does not delete the stored data permanently from the database. We use the WHERE clause with the DELETE command to select specific rows from the table.

Syntax of DELETE Command: DELETE FROM Table_Name WHERE condition;



```
Autocommit Display 10
DELETE FROM Bills
WHERE BillID = 1;
```

Results Explain Describe Saved SQL History

1 row(s) deleted.

0.00 seconds

Conclusion: In conclusion, Data Definition Language (DDL) and Data Manipulation Language (DML) are two important components of a database management system. Together, DDL and DML provide the necessary tools for managing and manipulating data in a database. They allow for the creation and modification of the database schema and the insertion, updating, and deletion of data. By using these tools effectively, users can ensure that the data in the database is accurate, consistent, and organized.

Findings/Learnings: By designing and implementing a well-structured hospital model in a DBMS, hospital staff can more easily and accurately manage patient data and provide efficient healthcare services.

Experiment 8

Program Objective: To Implement Simple Queries

Program theory:

The SQL SELECT statement is used to retrieve data from one or more tables in a database.

The basic syntax for the SELECT statement is:

SELECT column1, column2, ...

FROM table_name;

Performing Simple queries:

- SQL to count number of Bills from Bills table where amount is greater than 5000

The screenshot shows a SQL query execution interface. At the top, there is a checkbox for 'Autocommit' which is checked, and a 'Display' dropdown menu set to '10'. Below this, the SQL query is entered: `SELECT COUNT(*) FROM Bills WHERE totalcost > 4000;`. The query is executed, and the results are displayed in a table with one row: `COUNT(*)` with the value `2`. Below the table, it says '1 rows returned in 0.00 seconds' and there is a 'CSV Export' link.

COUNT(*)
2

This query uses the COUNT function to count the number of rows that match the condition amount > 5000 in the bill table. The * symbol is used as a placeholder to count all columns in the table.

- SQL query to select the Patient name and address from the patient table where the patient name starts with "r":

The screenshot shows a SQL query execution interface. At the top, there is a checkbox for 'Autocommit' which is checked, and a 'Display' dropdown menu set to '10'. Below this, the SQL query is entered: `SELECT Patientname, Address FROM Patients WHERE Patientname LIKE 'r%';`. The query is executed, and the results are displayed in a table with two columns: 'PATIENTNAME' and 'ADDRESS'. The first row shows 'Lalu Yadav' and 'Bihar'. Below the table, it says '1 rows returned in 0.00 seconds' and there is a 'CSV Export' link.

PATIENTNAME	ADDRESS
Lalu Yadav	Bihar

This query uses the LIKE operator to match the patient's name with the pattern 'r%'. The '%' symbol is a wildcard character that matches any number of characters. Therefore, the condition name LIKE 'r%' will match any patient name that starts with the letter 'r'. The SELECT statement is used to select the name and address columns from the patient table for the matched rows.

- Count number of patients whose name consists of 'a' Letter

Home > SQL > **SQL Commands**

☒ Autocommit Display 10

```
SELECT COUNT(*)
FROM Patient
WHERE patientName Like 'a%';
```

Results Explain Describe Saved SQL History

COUNT(*)
3

1 rows returned in 0.01 seconds [CSV Export](#)

- Select those values from department table where Hod name starts from 'a'.

Home > SQL > **SQL Commands**

☒ Autocommit Display 10

```
SELECT *
FROM Department
WHERE hodName LIKE 'A%';
```

Results Explain Describe Saved SQL History

DEPTNO	HODNAME	DEPTNAME
101	Anil Kumar Jha	Neurology

- Get the ReportID who have been prescribed a particular medicine:

Home > SQL > **SQL Commands**

☒ Autocommit Display 10

```
SELECT ReportID FROM Medicine WHERE MedicineName = 'Aspirin';
```

Results Explain Describe Saved SQL History

REPORTID
1

1 rows returned in 0.01 seconds [CSV Export](#)

- Select from the billing table where the total cost of the bill is greater than 10000

Home > SQL > **SQL Commands**

☒ Autocommit Display 10

```
SELECT *
FROM Billings
WHERE totalCharge > 10000;
```

Results Explain Describe Saved SQL History

BILLID	PAT_NAME	DUE DATE	TOTALCHARGE	MODEFPAY	ACCOUNTNO
405	Nalin	19-MAR-22	15000	Paytm	11000763528
407	Nishant	15-MAR-22	15700	Phonepe	110001228

2 rows returned in 0.02 seconds [CSV Export](#)

- Get ReportID of patients who have leg injury

Home > SQL > SQL Commands

☒ Autocommit Display 10 ▼

```
SELECT ReportID FROM PrescriptionReport WHERE ReportDescription = 'Leg Injury';
```

Results Explain Describe Saved SQL History

REPORTID
1

1 rows returned in 0.00 seconds [CSV Export](#)

Conclusion: From this experiment we concluded that SELECT statement is used for retrieving data from the database by applying certain conditions onto it.

Findings/ Learnings: We have learned that the SELECT statement is probably the most important SQL command. It's used to return results from our database(s) and no matter how easy that could sound, it could be really very complex.

Experiment 9

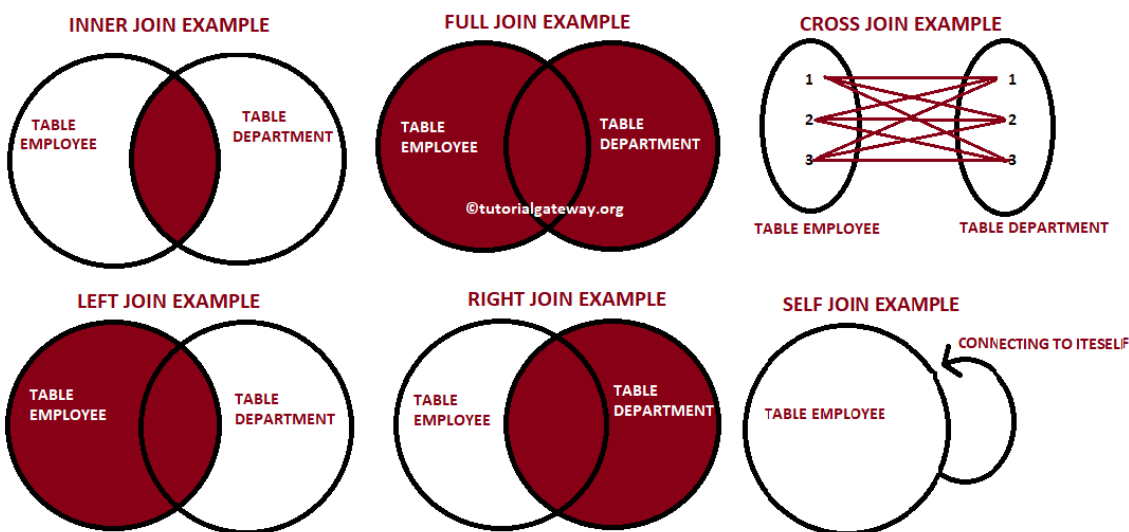
Program Objective: Implementation of Joins

Program theory:

A SQL Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are:

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL JOIN

The simplest Join is INNER JOIN.



1. INNER JOIN: The INNER JOIN keyword selects all rows from both the tables as long as the condition satisfies. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be same.

Syntax:

```
SELECT tableA.column1,  
tableA.column2,tableB.column1, ....  
FROM tableA  
INNER JOIN tableB  
ON tableA.matching_column =  
tableB.matching_column;
```

Home > SQL > SQL Commands		
<input checked="" type="checkbox"/> Autocommit	Display	10
<pre>SELECT BILLINGS.PAT_NAME, COST.TESTNO, COST.TESTCOST FROM BILLINGS INNER JOIN COST ON BILLINGS.BILLID = COST.BILLNO;</pre>		
Results Explain Describe Saved SQL History		
PAT_NAME	TESTNO	TESTCOST
Nishant	501	2500
Manish	503	1500
Nalin	505	1000
Mukesh	508	900
Mukul	510	500

2. LEFT JOIN: This join returns all matching rows for the table on the right side of join. The rows for which there is no matching row on right side, the result-set will contain null.

Syntax:

```
SELECT tableA.column1,tableA.column2,tableB.column1, ....
FROM tableA
LEFT JOIN tableB
ON tableA.matching_column = tableB.matching_column;
```

Home > SQL > **SQL Commands**

☒ Autocommit Display 10 ▼

```
SELECT DEPARTMENT.DEPTNAME, CONSIST_OF.NURSENAME
FROM DEPARTMENT
LEFT JOIN CONSIST_OF
ON DEPARTMENT.DEPTNO = CONSIST_OF.DEPT_ID;
```

Results Explain Describe Saved SQL History

DEPTNAME	NURSENAME
Neurology	Dixita
Emergency	Deeksha
Inpatient	Anushka
Outpatient	Anshika
Pharmacy	Deepanshi

5 rows returned in 0.01 seconds [CSV Export](#)

3. RIGHT JOIN: RIGHT JOIN is similar to LEFT JOIN. This join returns All the rows of the table on the right side of the join and matching rows for the table on the left side of join. The rows for which there is no matching row on left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN.

Syntax:

```
SELECT tableA.column1,
tableA.column2,tableB.column1,
.....
FROM tableA
RIGHT JOIN tableB
ON tableA.matching_column
= tableB.matching_column;
```

Home > SQL > **SQL Commands**

☒ Autocommit Display 10 ▼

```
SELECT WARD.WARDNAME, WARD.WARDCAP, HAS.PATNAME, HAS.EMP_BED
FROM WARD
RIGHT JOIN HAS
ON WARD.WARDID = HAS.WARD_NO;
```

Results Explain Describe Saved SQL History

WARDNAME	WARDCAP	PATNAME	EMP_BED
Sister Nivadita	100	Mukul	3
Kalpna Chawla	150	Mahak	4
Aryabhata	90	Pankaj	4
C.V. Raman	110	Mudit Raj	4
Ramanujan	130	Nitin	4

5 rows returned in 0.02 seconds [CSV Export](#)

4. FULL JOIN: FULL JOIN creates the result-set by combining result of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both the tables. The rows for which there is no matching, the result-set will contain NULL values.

Syntax:

```
SELECT table1.column1, table1.column2, table2.column1, ....
FROM table1
```

FULL JOIN table2

ON table1.matching_column = table2.matching_column;

Home > SQL > SQL Commands

☒ Autocommit Display 10

```
SELECT TEST.TESTNAME, TEST.TESTCOST, UNDERGOES.PATNAME, UNDERGOES.PATNO
FROM TEST
FULL JOIN UNDERGOES
ON TEST.TESTID = UNDERGOES.PATNO;
```

Results Explain Describe Saved SQL History

TESTNAME	TESTCOST	PATNAME	PATNO
KFT	900	-	-
X-Ray	1500	-	-
Ultra Sound	1000	-	-
LFT	500	-	-
CT Scan	2500	-	-
-	-	Nitin	303
-	-	Mahak	302
-	-	Pankaj	304
-	-	Mudit Raj	305

9 rows returned in 0.02 seconds [CSV Export](#)

Conclusion: We concluded that Joins are important statements in database because they let us combine results and at the same time filter out (or add in) information as needed. SQL tends to be a language of refinements, and we will probably use joins in efforts to refine results rather than to create initial SQL statements.

Findings/ learnings:

From this experiment we get to learn about different joins in SQL i.e., inner join, left join, right join, full join and also implemented in our database using oracle 10g express. Hence, all the joins were understood and it was implemented in our database for effective database maintenance.

Experiment 10

Program Objective: Implementation of Aggregate functions, Date functions and Time functions

Program theory:

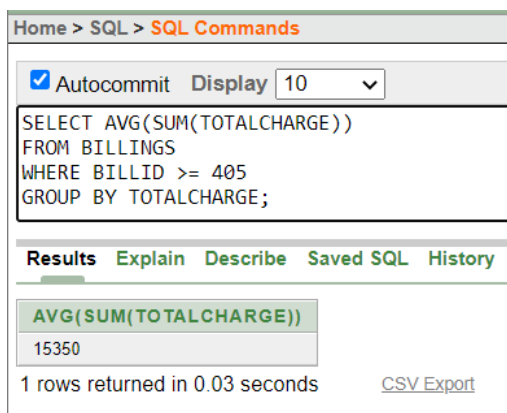
AGGREGATE FUNCTION:

Aggregate functions return a single value based on groups of rows, rather than single value for each row. You can use Aggregate functions in select lists and in ORDER BY and HAVING clauses. They are commonly used with the GROUP BY clause in a SELECT statement, where Oracle divides the rows of a queried table or view into groups.

The important Aggregate functions are:

- Avg
- Sum
- Max
- Min
- Count

1) AVG



Home > SQL > SQL Commands

☒ Autocommit Display 10

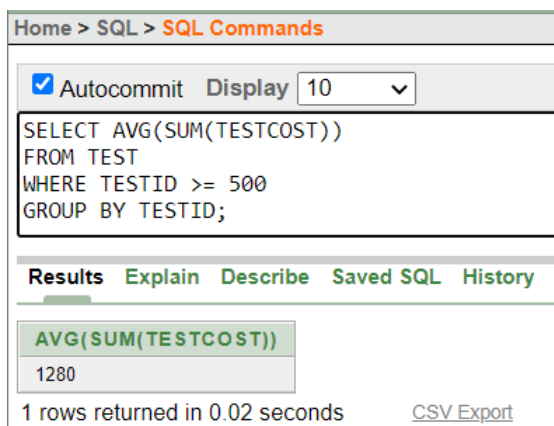
```
SELECT AVG(SUM(TOTALCHARGE))  
FROM BILLINGS  
WHERE BILLID >= 405  
GROUP BY TOTALCHARGE;
```

Results Explain Describe Saved SQL History

AVG(SUM(TOTALCHARGE))
15350

1 rows returned in 0.03 seconds [CSV Export](#)

- Returns the average value of the expression.



Home > SQL > SQL Commands

☒ Autocommit Display 10

```
SELECT AVG(SUM(TESTCOST))  
FROM TEST  
WHERE TESTID >= 500  
GROUP BY TESTID;
```

Results Explain Describe Saved SQL History

AVG(SUM(TESTCOST))
1280

1 rows returned in 0.02 seconds [CSV Export](#)

2) SUM

- Returns the sum of the value of the expression.

Home > SQL > **SQL Commands**

☒ Autocommit Display 10 ▼

```
SELECT SUM(WARDCAP)
FROM WARD
WHERE WARDID > 203;
```

Results Explain Describe Saved SQL History

SUM(WARDCAP)
240

1 rows returned in 0.00 seconds [CSV Export](#)

3) MAX

- Returns the maximum value of the selected column/expression.

Home > SQL > **SQL Commands**

☒ Autocommit Display 10 ▼

```
SELECT MAX(TESTCOST)
FROM TEST;
```

Results Explain Describe Saved SQL History

MAX(TESTCOST)
2500

1 rows returned in 0.00 seconds [CSV Export](#)

4) MIN

- Returns the minimum value of the selected column/expression.

Home > SQL > **SQL Commands**

☒ Autocommit Display 10 ▼

```
SELECT MIN(TOTALCHARGE)
FROM BILLINGS;
```

Results Explain Describe Saved SQL History

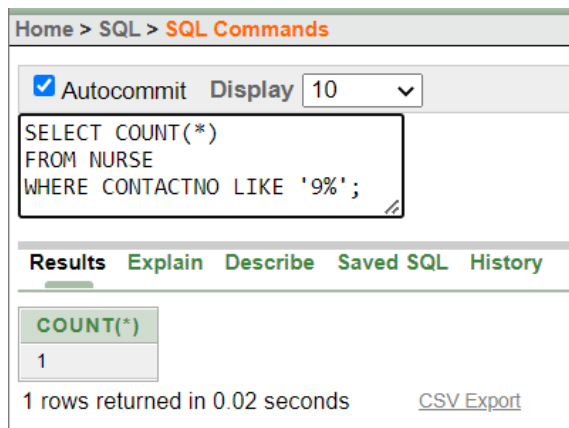
MIN(TOTALCHARGE)
2500

1 rows returned in 0.00 seconds [CSV Export](#)

5) COUNT

- Returns the number of rows in the query.

NOTE: If we specify the expression then count ignores NULLS. If we specify the asterisk(*), this function returns the count of all the rows, including duplicates and NULLS. COUNT never returns NULL.



Home > SQL > SQL Commands

☒ Autocommit Display 10

```
SELECT COUNT(*)  
FROM NURSE  
WHERE CONTACTNO LIKE '9%';
```

Results Explain Describe Saved SQL History

COUNT(*)
1

1 rows returned in 0.02 seconds [CSV Export](#)

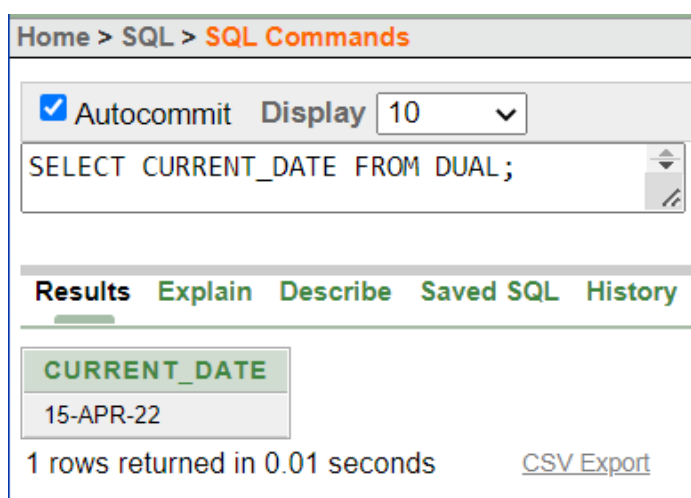
DATE AND FUNCTION:

The date and time function allows us to handle date and time data effectively. While working with the database, the format of date and time functions should be matched while inserting data into the table. In this article, we'll go through the date and time functions in depth. SQL Server has a number of built-in functions that assist us in filtering useful data from a vast quantity of data. It is also useful in designing and maintaining a large-scale database.

We have a variety of data types in SQL Server that can be used as the date in our table. The most popular format of date is 'YYYY-MM-DD' and 'DD-MM-YYYY.'. In some circumstances, we also need to save time with the date in our database. In these cases, we require tools to access the time and date separately. This is where SQL Server's time and functions are useful. It is also recommended that the beginner be cautious when using date or time in the database, as these are prone to throwing exceptions if not handled correctly.

CURRENT_DATE

This function is used to return the current date of the system.



Home > SQL > SQL Commands

☒ Autocommit Display 10

```
SELECT CURRENT_DATE FROM DUAL;
```

Results Explain Describe Saved SQL History

CURRENT_DATE
15-APR-22

1 rows returned in 0.01 seconds [CSV Export](#)

CURRENT_TIMESTAMP

- Return the current date and time with time zone in the session time.

Home > SQL > **SQL Commands**

☒ Autocommit Display 10

SELECT CURRENT_TIMESTAMP FROM DUAL;

Results Explain Describe Saved SQL History

CURRENT_TIMESTAMP
15-APR-22 04.32.36.065000 AM +00:00

1 rows returned in 0.01 seconds [CSV Export](#)

DBTIMEZONE

- Returns the time zone in which database is set.

Home > SQL > **SQL Commands**

☒ Autocommit Display 10

SELECT DBTIMEZONE FROM DUAL;

Results Explain Describe Saved SQL History

DBTIMEZONE
+00:00

1 rows returned in 0.02 seconds [CSV Export](#)

LOCALTIMESTAMP

Home > SQL > **SQL Commands**

☒ Autocommit Display 10

SELECT LOCALTIMESTAMP FROM DUAL;

Results Explain Describe Saved SQL History

LOCALTIMESTAMP
15-APR-22 04.35.47.241000 AM

1 rows returned in 0.01 seconds [CSV Export](#)

- Return a **TIMESTAMP** value that represents the current date and time in the session time zone.

SESSIONTIMEZONE

- Get the session time zone.

Home > SQL > **SQL Commands**

☒ Autocommit Display 10

SELECT SESSIONTIMEZONE FROM DUAL;

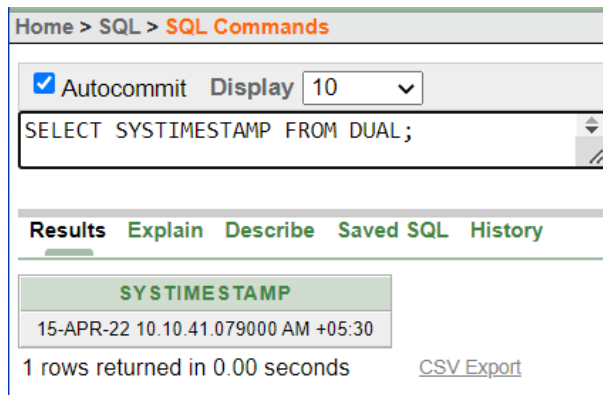
Results Explain Describe Saved SQL History

SESSIONTIMEZONE
+00:00

1 rows returned in 0.00 seconds [CSV Export](#)

SYSTIMESTAMP

Returns all the details about the timestamp of the system.



The screenshot shows a web-based SQL interface. At the top, there is a breadcrumb trail: "Home > SQL > SQL Commands". Below this, there is a toolbar with a checked "Autocommit" checkbox and a "Display" dropdown menu set to "10". The main text area contains the SQL query: "SELECT SYSTIMESTAMP FROM DUAL;". Below the query area, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History". The "Results" tab is active, showing a table with one column named "SYSTIMESTAMP" and one row containing the value "15-APR-22 10.10.41.079000 AM +05:30". At the bottom, it states "1 rows returned in 0.00 seconds" and provides a "CSV Export" link.

SYSTIMESTAMP
15-APR-22 10.10.41.079000 AM +05:30

1 rows returned in 0.00 seconds [CSV Export](#)

Conclusion:

We concluded that the date and time functions in DBMS are quite useful to manipulate and store values related to date and time. Using the date and time functions we can find the date today; we can check the date after some given time and so on. We also have the functionality to dissect the given date into various components like the month, the year, or just the day or a combination of all. With this added functionality, we are able to utilize SQL for large-scale database designing and maintaining.

Findings/ learnings:

From this experiment we learned about different aggregate functions like Avg, Sum, Max, Min, Count, etc and also about various Date and Time functions which return the details regarding time and date of the system according to the performed query.

Experiment 11

Program Objective: Implement Grant and Revoke Commands, Views and Triggers on Hospital Management System

Program theory:

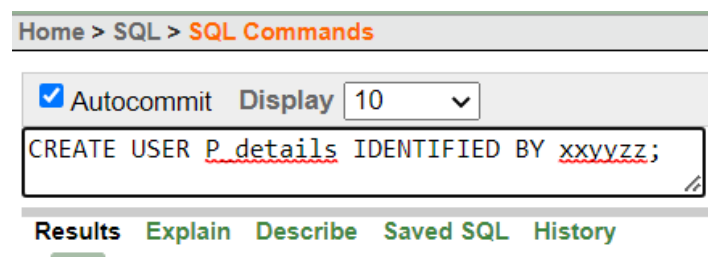
- **GRANT**

DCL commands are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE. Only Database Administrator's or owners of the database object can provide/remove privileges on a database object. SQL GRANT is a command used to provide access or privileges on the database objects to the users. Using Grant, we can give access rights to users like ALL, EXECUTE AND SELECT with specified roles defined to them.

SYNTAX:

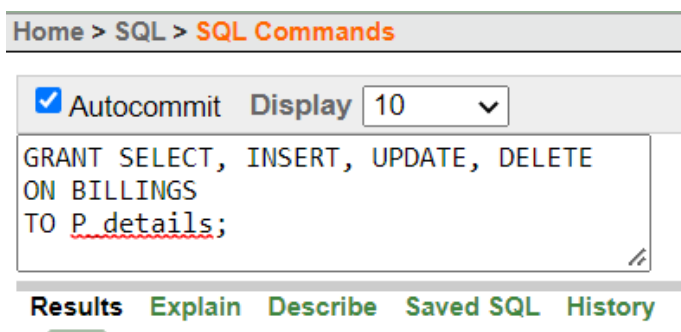
GRANT privilege_name ON object_name

TO {user_name |PUBLIC |role_name} [WITH GRANT OPTION];



User created.

0.30 seconds



Statement processed.

0.05 seconds

- **REVOKE**

SQL Revoke is used to remove the permissions or privileges of a user on database objects set by the Grant command. Revoke command withdraw user privileges on database objects if any granted. It does operations opposite to the Grant command. When a privilege is revoked from a particular user U, then the privileges granted to all other users by user U will be revoked. If access for one user is removed; all the particular permissions provided by that users to others will be removed.

SYNTAX:

REVOKE privileges ON object FROM user

Home > SQL > **SQL Commands**

☒ Autocommit Display 10 ▼

```
REVOKE ALL
ON BILLINGS
FROM P_details;
```

Results Explain Describe Saved SQL History

Statement processed.

0.05 seconds

- **VIEWS**

Views in SQL are kind of virtual tables. A view also has rows and columns as they are in a real table in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition. In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

SYNTAX:

CREATE VIEW view_name AS SELECT columns

FROM tables WHERE conditions;

Home > SQL > **SQL Commands**

☒ Autocommit Display 10 ▼

```
CREATE VIEW P_Details AS
SELECT PATIENTID, PATIENTNAME
FROM PATIENT
WHERE PATIENTAGE >= 19;
```

Results Explain Describe Saved SQL History

View created.

0.16 seconds

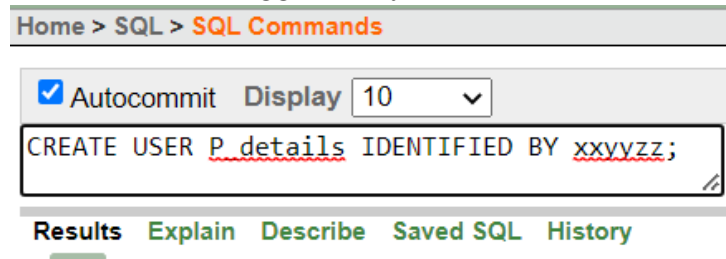
- **TRIGGERS**

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated. A SQL trigger is a database object which fires when an event occurs in a database. We can execute a SQL query that will "do something" in a database when a change occurs on a database table such as a record is inserted or updated or deleted. For example, a trigger can be set on a record insert in a database table.

There are two types of trigger in SQL- DDL TRIGGERS and DML TRIGGERS which can further be classified into 12 types.

SYNTAX:

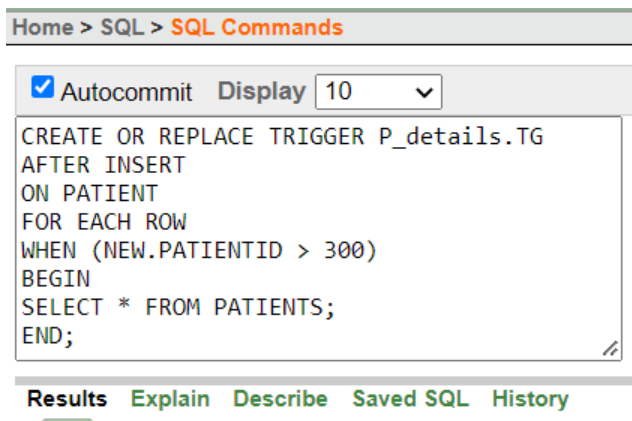
```
create trigger [trigger_name] [before | after]
{insert | update | delete} on [table_name]
[for each row] [trigger_body]
```



The screenshot shows a SQL Command window with the title 'Home > SQL > SQL Commands'. It has a toolbar with 'Autocommit' checked and 'Display' set to 10. The command text area contains 'CREATE USER P_details IDENTIFIED BY xxvzz;' with red squiggly lines under the username and password. Below the text area are tabs: 'Results' (selected), 'Explain', 'Describe', 'Saved SQL', and 'History'.

User created.

0.30 seconds



The screenshot shows a SQL Command window with the title 'Home > SQL > SQL Commands'. It has a toolbar with 'Autocommit' checked and 'Display' set to 10. The command text area contains the following SQL code: 'CREATE OR REPLACE TRIGGER P_details.TG AFTER INSERT ON PATIENT FOR EACH ROW WHEN (NEW.PATIENTID > 300) BEGIN SELECT * FROM PATIENTS; END;'. Below the text area are tabs: 'Results' (selected), 'Explain', 'Describe', 'Saved SQL', and 'History'.

Trigger created.

0.02 seconds

Conclusion:

We concluded that the owner of the database or the administrator of the database can provide or remove the privileges given to the database objects. So, the DCL commands are used to ensure the security of a database in an environment where multiple databases or various users are present by controlling the privileges given to the users.

Findings/ learnings:

From this experiment we learned that Create User Statement only creates a new user but does not grant any privileges to the user account. Therefore, to grant privileges to a user account, the GRANT statement is used. The Revoke statement is used to revoke some or all of the privileges which have been granted to a user in the past and a trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.