

DELHI TECHNOLOGICAL UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CO 202: DATABASE MANAGEMENT SYSTEMS

PRACTICAL FILE

Submitted To:

Ms. Indu Singh

Assistant Professor

Department of CSE, DTU

Submitted By:

Nikhil Mishra

2K19/CO/250

A4 Batch Group 1

TABLE OF CONTENTS

S.NO	Title	Date	Page No.
1.	Introduction to SQL, Database, and Database Management System.	January 18, 2021	4
2.	Introduction to Various software of DBMS and Database Languages.	January 25, 2021	6
3.	Introduction to ER Diagram and Symbol table.	February 8, 2021	9
4.	Introduction to Different types of constraints in SQL	February 22, 2021	13
5.	Case Study 1: Railway Management System	March 12, 2021	16
6.	Creating Entity-Relationship Diagram, Entity Tables and Relationship Tables	Match 12, 2021	18
7.	Implementing DDL and DML commands.	March 22, 2021	21
8.	Performing Simple Queries.	March 29, 2021	32
9.	Implementation of Simple Joins	March 31, 2021	36

10.	Implementing Aggregate Function, Date and Time Function	April 27, 2021	40
11.	Implementing Triggers, Views, Grant and Revoke commands	April 29,2021	48

EXPERIMENT 1

AIM: Introduction to SQL, DATABASE and DATABASE MANAGEMENT SYSTEM.

THEORY

SQL:

- SQL stands for Structured Query Language. It allows us to access and manipulate databases. It is the standard database language for Relational Database Management Systems (RDBMS). SQL is very convenient to use. It can do many operations like: execute queries, retrieve data, insert, update and delete records. SQL contains multiple commands that allow it perform various types of operations on a database. Some of the common commands are:
 - CREATE to create a new table or database.
 - ALTER for alteration.
 - Truncate to delete data from the table.
 - DROP to drop a table.
 - RENAME to rename a table.
 - INSERT to insert a new row.
 - UPDATE to update an existing row.
 - DELETE to delete a row.
 - MERGE for merging two rows or two tables.

DATABASE:

- Database is the collection and organization of inter-related data in electronic form. The data should be efficiently arranged so that easy insertion, deletion, presentation and retrieval of data is possible. For example: a school creates a database to organize the details of all students, teachers and staff in a school. This enables efficient record-keeping of everyone. The database can be further queried to extract information of required students or teachers as desired. All these are possible due to Database Management Systems.

DATABASE MANAGEMENT SYSTEM:

- A Database Management System is a software that is designed to help create and maintain databases. It allows for the efficient storage of data. Popular DBMS software's are My SQL, Oracle, dBASE, FoxPro etc. Some of its common functions are:
 - Data Dictionary Management: DBMS stores description about the data in the data dictionary and uses it to look up the required structures and relationships.

- Data Storage Management: It efficiently stores all data in the database and manages it by itself. The user doesn't need to know how the data is being stored.
- Efficient Query Processing: DBMS allows efficient query processing as queries can be executed in simple English-like syntaxes.
- Restrict Unauthorized Access: It uses an Access Control Matrix to manage to maintain data integrity and prevent any intrusions.

CONCLUSION:

This experiment introduces us to the concepts of database, DBMS and SQL and their respective advantages and disadvantages. It also gives us a small outlook on SQL commands.

EXPERIMENT 2

AIM: Introduction to various Database Management System and Database Language

THEORY

Database Management System Software:

Database Management System Software (DBMS) is a software that allows users to create, present and maintain databases. It allows efficient manipulation of data. There are various DBMS softwares. Some of them are:

- **My SQL:**
It is an open source RDBMS software that uses structured query language processing. It is used by major companies such as Facebook, Google, Adobe, Alcatel Lucent and Zappos to power their high volume websites. The major features of this software is: High-speed data processing, use of triggers increases productivity, with rollback and commit helps in data recovery if required.
- **Oracle:**
It is the most widely used DBMS software in the world. It supports multiple Linux, Unix and Windows versions. The latest version, Oracle 12c supports cloud services. It is a relational database management software. It is secured, occupies less space, supports large databases, and reduces CPU time to process data.
- **Microsoft Access:**
It is a DBMS software that uses Graphical User Interface (GUI) and software development tools. It is considered ideal for beginners to learn basics of database due to its use of GUI. It is used by most e-commerce websites. It works only on Microsoft Windows.
- **dBASE:**
It was one of the first DBMS software to be offered for microcomputers. The system includes core database design, a query system, forms engine and a programming language that interconnects all these components.
- **SolarWinds Database Performance Analyzer:**
It is the database management software that can perform SQL query performance monitoring, analysis, and tuning. It supports cross-platform database performance tuning and optimization. Some of its features are: has the features of Machine Learning, Cross-Platform Database Support, Expert Tuning Advisors, Cloud Database Support, and Automation Management API, etc.
- **Microsoft SQL Server:**

Developed by Microsoft in 1989, it works on both Linux and Windows Operating Systems. The language used is Assembly C, Linux, C++ for writing it. It is compatible with Oracle, provides efficient management of workload and allows multiple users to use the same database.

- **SQLite:**

It is used as a database system for mobiles. It is coded in C language. It can work on Linux, Windows, and Mac operating systems. It does not need much space hence, it can be used for storing small to medium size websites. It is fast and does not need to set up.

Apart from these there are various other DBMS software that follow hierarchical, relational, network or object oriented model.

Database Languages:

Database Languages are used to perform various types of operations on a database. It is used to define and manipulate a database. It is of five major types. They are:

- **Data Definition Language:** It is referred to as DDL. It is used to describe the database structure and pattern. It is used to create the skeleton of the database. The DDL commands are as follows.
 - **Create:** It is used to create objects in the database.
 - **Alter:** It is used to alter the structure of the database.
 - **Drop:** It is used to delete objects from the database.
 - **Truncate:** It is used to remove all records from a table.
 - **Rename:** It is used to rename an object.
 - **Comment:** It is used to comment on the data dictionary.
- **Data Manipulation Language:** It is referred to as DML. It is used for accessing and manipulating data in a database. The DML commands are as follows.
 - **Select:** It is used to retrieve data from a database.
 - **Insert:** It is used to insert data into a table.
 - **Update:** It is used to update existing data within a table.
 - **Delete:** It is used to delete all records from a table.
 - **Merge:** It performs UPSERT operation, i.e., insert or update operations.
 - **Call:** It is used to call a structured query language or a Java subprogram.
 - **Explain Plan:** It has the parameter of explaining data.
 - **Lock Table:** It controls concurrency.

- **Data Control Language:** It is referred to as DCL. It is used to retrieve the stored data. The DCL commands are as follows.
 - **Grant:** It is used to give user access privileges to a database.
 - **Revoke:** It is used to take back permissions from the user.
 - **Analyze:** It is used to provide statistical information of relations.
 - **Audit:** It is used to track occurrence of specific SQL statements during specified user sessions.
- **Transaction Control Language:** It is referred to as TCL. It is used to run the changes made by the DML statement. The TCL commands are as follows.
 - **Commit:** It is used to save the transaction on the database.
 - **Rollback:** It is used to restore the database to original since the last Commit.
 - **Savepoint:** It is used to rollback transactions making savepoints within it.
- **Session Control Language:** It is referred to as SCL. It is used to dynamically manage the properties of a session. The SCL commands are as follows.
 - **Alter Session:** It is used to modify conditions or parameters affecting database connections.
 - **Set Role:** It is used to enable or disable the roles that are currently enabled for the session.

CONCLUSION:

This experiment introduced us to various DBMS software and help gain an insight about various Data Languages. Various SQL commands were briefly discussed.

EXPERIMENT 3

AIM: Introduction to ER Diagram and Symbol Table.

THEORY

ER Diagram:

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. ER models are required to properly implement a database.

By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases. ER diagrams are used to sketch out the design of a database. It helps us describe the logical structure of a database. It helps us analyze data requirements systematically.

The major reasons for constructing ER diagrams are:

- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ER Diagram allows you to communicate with the logical structure of the database to users

Advantages of using an ER diagram:

- **Conceptually it is very simple:** ER model is very simple because if we know relationship between entities and attributes, then we can easily draw an ER diagram.
- **Better visual representation:** ER model is a diagrammatic representation of any logical structure of database. By seeing ER diagram, we can easily understand relationship among entities and relationship.
- **Effective communication tool:** It is an effective communication tool for database designer.

- **Highly integrated with relational model:** ER model can be easily converted into relational model by simply converting ER model into tables.
- **Easy conversion to any data model:** ER model can be easily converted into another data model like hierarchical data model, network data model and so on.

Disadvantages of using an ER diagram:

- **Loss of information content:** Some information be lost or hidden in ER model
- **Limited relationship representation:** ER model represents limited relationship as compared to another data models like relational model etc.
- **No representation of data manipulation:** It is difficult to show data manipulation in ER model.
- **Popular for high level design:** ER model is very popular for designing high level design

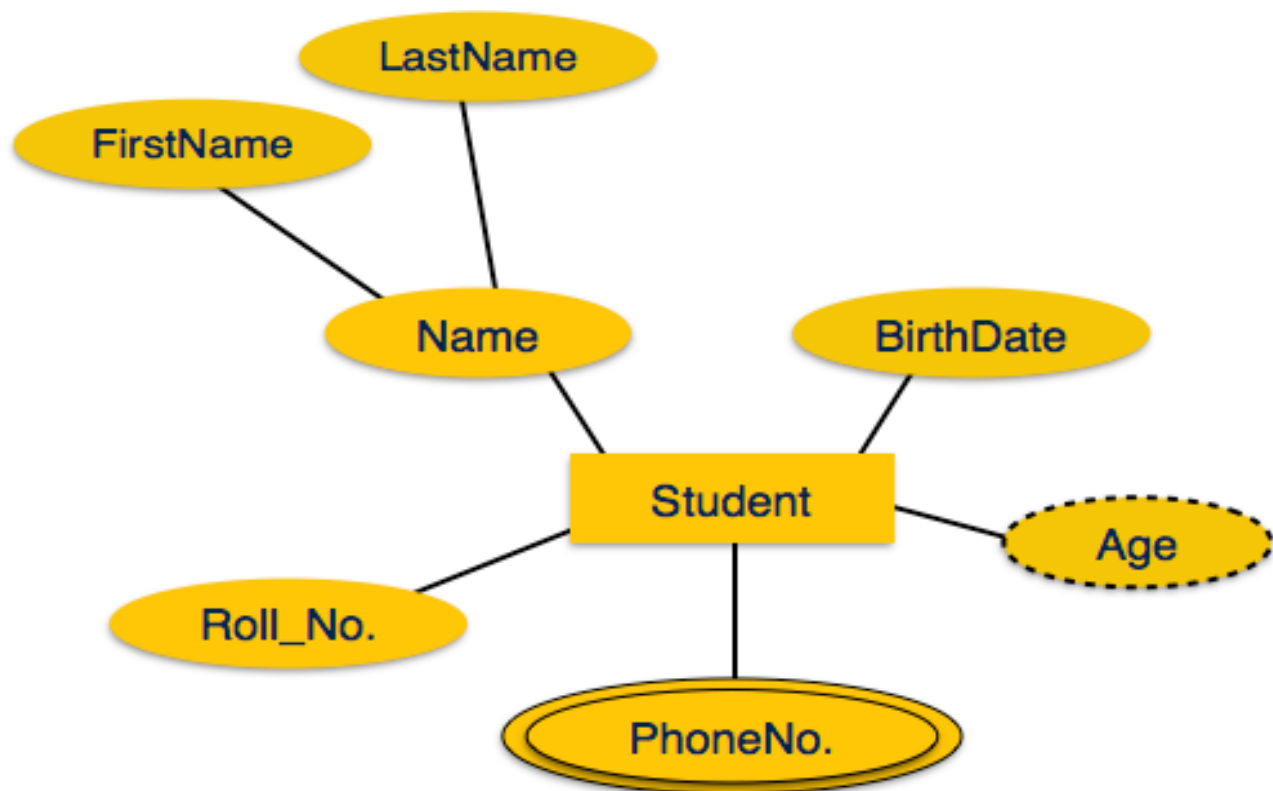


Figure: Basic Entity-Relationship Diagram

Symbol Table:

Symbol table is an important data structure created and maintained by compilers in order to store information about the occurrence of various entities such as variable names, function names, objects, classes, interfaces, etc. Symbol table is used by both the analysis and the synthesis parts of a compiler. Symbol table is used to store the information about the

occurrence of various entities such as objects, classes, variable name, interface, function name etc. it is used by both the analysis and synthesis phases.

A symbol table may serve the following purposes depending upon the language in hand:

- To store the names of all entities in a structured form at one place.
- To verify if a variable has been declared.
- To implement type checking, by verifying assignments and expressions in the source code are semantically correct.
- To determine the scope of a name

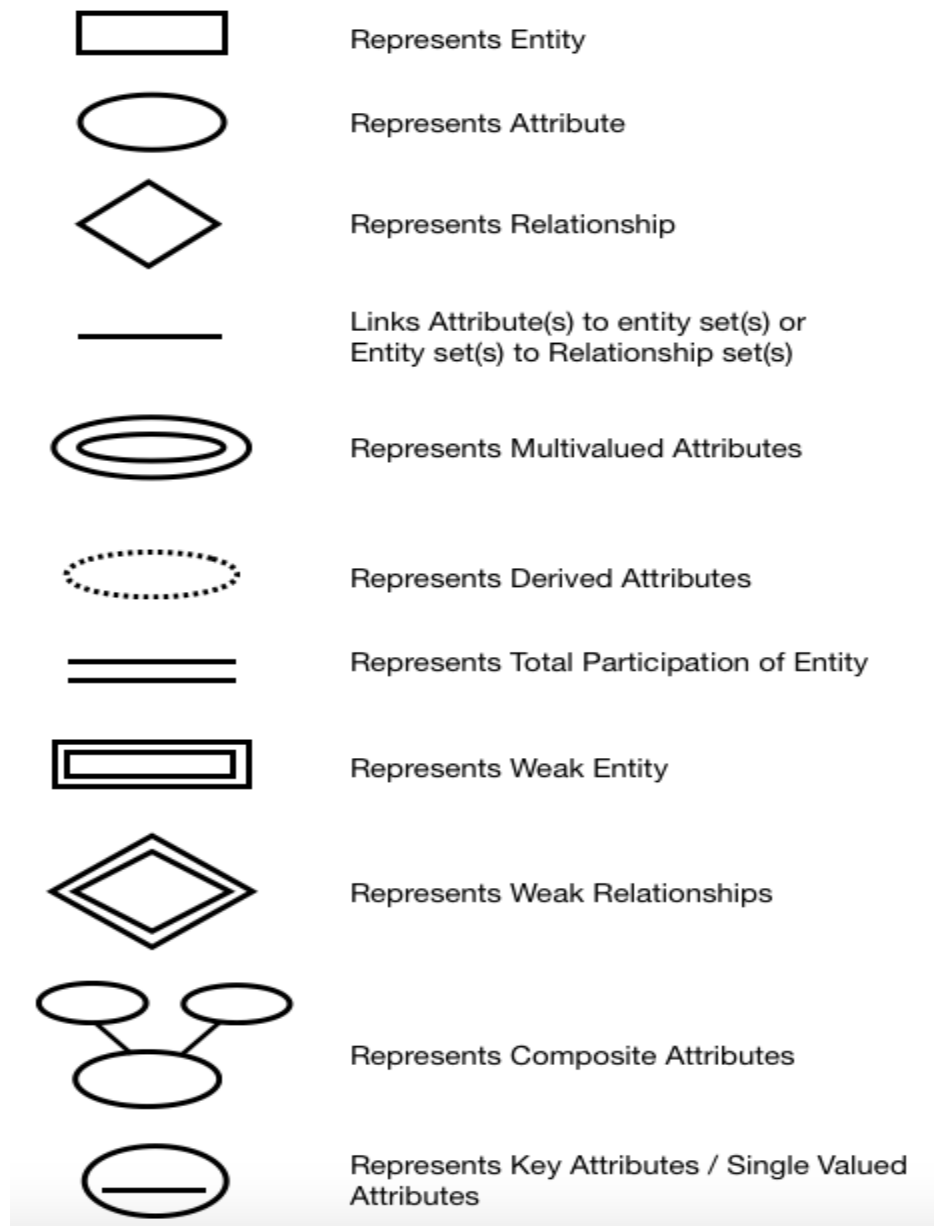


Figure: Symbols used in ER diagram

CONCLUSION:

In this experiment, the concept of ER Diagrams and Symbol Table was introduced. The various symbols used to represent various aspects of an ER diagram was explained.

EXPERIMENT 4

AIM: Introduction to various constraints in SQL.

THEORY

Constraints are the set of rules that can be enforced on the attributes in a relation. They are used to limit the range or type of data that can be entered in the relation. They can be applied to individual columns or the whole relation. If the constraints are violated, the said data is not entered into the relation. Any entry made into a relation must satisfy the specified constraints. The most common constraints are:

- **NOT NULL Constraint:**

It is used to ensure that any column does not except NULL values. For example: In a student table, the roll number field cannot be left NULL. In other words, every student must have a roll number and so the database can have a constraint in that field so that it does not accept null values.

```
CREATE TABLE Student (  
    Roll_No. int NOT NULL,  
    Name varchar(255) NOT NULL,  
    Address varchar(255),  
    Age int  
);
```

- **UNIQUE Constraint:**

It is used to ensure that all values in a column are unique, i.e. no value should be repeated. For example: In an students table, each students' roll number should be unique.

```
CREATE TABLE Students (  
    Roll_No int NOT NULL UNIQUE,  
    Name varchar (255) NOT NULL,  
    Address varchar(255),  
    Age int  
);
```

- **DEFAULT Constraint:**

It is used to provide a default value to a column when none is specified. For example, country of students could be given a default value.

```
CREATE TABLE Students (  
    Roll_No. int NOT NULL UNIQUE,  
    Name varchar(255) NOT NULL,  
    Address varchar(255),  
    Age int,  
    Country varchar(255) DEFAULT 'India'  
);
```

- **PRIMARY KEY Constraint:**

It is used to uniquely identify each record in a database. For example: Roll Number of student in a student table.

```
CREATE TABLE Students (  
    Roll_No. int NOT NULL UNIQUE PRIMARY KEY,  
    Name varchar(255) NOT NULL,  
    Address varchar(255),  
    Age int,  
    Country varchar(255) DEFAULT 'India'  
);
```

- **CHECK Constraint:**

It is used to ensure that a certain condition is met when a record is entered into a relation. For example: students studying in a school should not be more than 25 years old.

```
CREATE TABLE Students (  
    Roll_No. int NOT NULL UNIQUE PRIMARY KEY,  
    Name varchar(255) NOT NULL,  
    Address varchar(255),  
    Age int NOT NULL CHECK (age<=25),  
    Country varchar(255) DEFAULT 'India'  
);
```

- **FOREIGN KEY Constraint:**

It is used to indicate a row that uniquely identifies a record in another table. For example: department id acts as foreign key in employees table which acts as primary key in departments table.

```
CREATE TABLE Employees (  
    emp_id int NOT NULL UNIQUE PRIMARY KEY,  
    Name varchar(255) NOT NULL,  
    salary int,  
    dept_id int,  
    FOREIGN KEY(dept_id) REFERENCES departments(dept_id)  
);
```

CONCLUSION:

As observed, SQL contains multiple constraints and their working principles has been explained in the above experiment. They all have specific purpose as seen.

EXPERIMENT 5

CASE STUDY: RAILWAY MANAGEMENT SYSTEM

PROBLEM STATEMENT:

- Railway Reservation System will maintain the information about Reservation like Passenger name, Id etc.
- Railway Reservation System will maintain the information about passenger like Personal details and Official detail.
- System will maintain the information about train like Train no., Train name.
- Railway Reservation System will maintain the information about the seat availability like Source station, Destination station, Date, Train no., Class, Arrival time.
- Railway Reservation System will maintain the information about the Account like, Date, PNR and Amount.
- Railway Reservation System will maintain the information about the Train for cancellation like PNR, No. of seat, Date.

ASSUMPTIONS:

- One Passenger can reserve N seats in one train.
- One Passenger can pay the Ticket Rate by only one account.
- One Passenger can cancel the N seats at one time.
- Passenger details can be Personal or Official.

ER DIAGRAM DESCRIPTION

➤ Identification of Entities:

1. PASSENGER (Personal Id, Name, Age, Sex, Phone Number, Address, Office Name, Office Address, Office Phone)
2. TRAIN (Train No., Train Name, Duration, First Station, Last Station)
3. RESERVATIONS (Reservation No., Passenger Id, Passenger Name, Train Number, Number of Seats, Reservation Date)
4. ACCOUNT (PNR, Account Holder Name, Date of Booking, Amount)
5. CANCELLATION (Cancellation No., PNR, Train No., Train Name, Class, Number of Seats, Reservation Date)
6. SEAT AVAILABILITY (Train No., Reservation Date, Class, Quota, Total Seats, Available Seats)
7. TICKET (Train No., Ticket Rate)

➤ **Identification of Relationships:**

1. RESERVATION : PASSENGER (N:1)
2. RESERVATION : TRAIN (N:M)
3. TRAIN : SEAT AVAILABILITY (N:M)
4. TRAIN : ACCOUNT (N:1)

CONCLUSION:

This experiment helped us formulate the problem of the case study and make certain assumptions that will be helpful in studying the problem. It also helped us identify the key entities and their respective attributes which can be used to create relations. Consequently, the key relationships between the relations have also been identified during the course of this experiment which will be further used to construct ER diagrams.

EXPERIMENT 6

AIM: Creating Entity-Relationship Diagram, Entity Tables and Relationship Tables

THEORY

Entity Relationship Diagrams are used to define the relationships between various entities. The Entities and Relationships had been identified in Experiment 5. As observed, there are seven key entities and four major relationships. They are listed below.

➤ **Identification of Entities:**

8. PASSENGER (Personal Id, Name, Age, Sex, Phone Number, Address, Office Name, Office Address, Office Phone)
9. TRAIN (Train No., Train Name, Duration, First Station, Last Station)
10. RESERVATIONS (Reservation No., Passenger Id, Passenger Name, Train Number, Number of Seats, Reservation Date)
11. ACCOUNT (PNR, Account Holder Name, Date of Booking, Amount)
12. CANCELLATION (Cancellation No., PNR, Train No., Train Name, Class, Number of Seats, Reservation Date)
13. SEAT AVAILABILITY (Train No., Reservation Date, Class, Quota, Total Seats, Available Seats)
14. TICKET (Train No., Ticket Rate)

➤ **Identification of Relationships:**

5. RESERVATION : PASSENGER (N:1)
6. RESERVATION : TRAIN (N:M)
7. TRAIN : SEAT AVAILABILITY (N:M)
8. TRAIN : ACCOUNT (N:1)

The relevant Entity Relationship Diagram has been shown below.

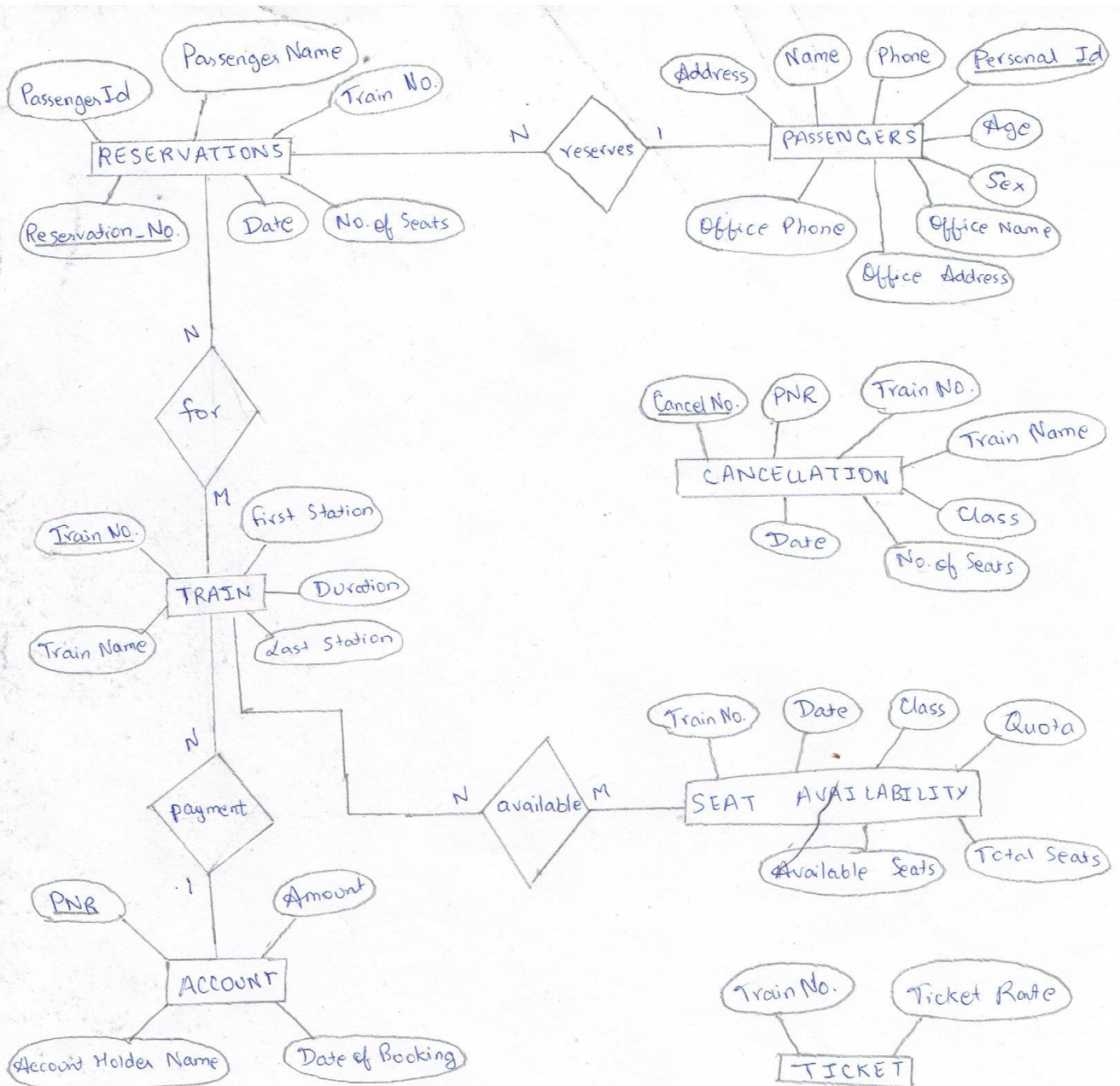


Figure: ER Diagram

CONCLUSION:

At the end of this experiment, we were able to fully identify the entities and relationships as well as construct a full ER Diagram using proper symbols. This experiment helped us simplify the case study and identify the number of table that need to be made, the attribute that they should contain as well as visualize the relationships between them.

EXPERIMENT 7

AIM: Implementing DDL and DML commands.

THEORY

DDL Commands:

DDL stands for Database Definition Language. These types of SQL commands are used to define the database schema. It is used to create or modify database objects and deals with the description of the database schema. The most commonly used DDL commands are:

1. **CREATE TABLE:** This command is used to create a new relation in the database.

Syntax: CREATE TABLE table_name

(Attribute1, Datatype,

Attribute2, Datatype...)

2. **ALTER TABLE:** This command is used to add, delete or manipulate attributes in a relation.

Syntax: ALTER TABLE table_name

ADD Attribute_Name datatype;

3. **DROP TABLE:** This command is used to remove or delete a relation from a database

Syntax: DROP TABLE table_name;

4. **TRUNCATE:** This command is used to erase complete data from an existing relation

Syntax: TRUNCATE TABLE table_name;

Implementing DDL Commands:

We will be implementing the CREATE and ALTER Table commands for this case study. The results are shown below.

CREATE:

- PASSENGERS Table:

```
CREATE TABLE PASSENGERS
(
  PERSONAL_ID INT NOT NULL PRIMARY KEY,
  NAME VARCHAR(50) NOT NULL,
  AGE INT NOT NULL CHECK(AGE<=120),
  SEX VARCHAR(10) NOT NULL,
  PHONE_NUMBER VARCHAR2(10),
  ADDRESS VARCHAR(50),
  OFFICE_NAME VARCHAR(50),
  OFFICE_ADDRESS VARCHAR(50),
  OFFICE_PHONE VARCHAR(10)
);|
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Table created.

5.22 seconds

|

- RESERVATIONS Table:

```
CREATE TABLE RESERVATIONS
(
  RESERVATION_NO INT NOT NULL PRIMARY KEY,
  PASSENGER_ID INT NOT NULL,
  PASSENGER_NAME VARCHAR(50) NOT NULL,
  TRAIN_NUM INT NOT NULL,
  NUMBER_OF_SEATS INT NOT NULL,
  RESERVATION_DATE DATE NOT NULL,
  FOREIGN KEY(TRAIN_NUM) REFERENCES TRAIN(TRAIN_NO)
);
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Table created.

0.04 seconds

- TRAIN Table:

```
CREATE TABLE TRAIN
(
  TRAIN_NO INT NOT NULL PRIMARY KEY,
  TRAIN_NAME VARCHAR(50) NOT NULL,
  DURATION INT,
  FIRST_STATION VARCHAR(50),
  LAST_STATION VARCHAR(50)
);
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Table created.

0.19 seconds

- TICKET Table:

```
CREATE TABLE TICKET
(
  TRAIN_NUMBER INT NOT NULL,
  TICKET_RATE INT NOT NULL,
  FOREIGN KEY(TRAIN_NUMBER) REFERENCES TRAIN(TRAIN_NO)
);
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Table created.

0.02 seconds

- ACCOUNT Table:

```
CREATE TABLE ACCOUNT
(
  PNR INT NOT NULL PRIMARY KEY,
  ACCOUNT_HOLDER_NAME VARCHAR(50) NOT NULL,
  DATE_OF_BOOKING DATE NOT NULL,
  AMOUNT INT NOT NULL
);
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Table created.

0.02 seconds

- CANCELLATION Table:

```
CREATE TABLE CANCELLATION
(
  CANCEL_NO INT NOT NULL PRIMARY KEY,
  PNR_NUM INT NOT NULL,
  TRAIN_NUM INT NOT NULL,
  TRAIN_NAME VARCHAR(50) NOT NULL,
  CLASS VARCHAR (15) NOT NULL,
  NUM_OF_SEATS INT NOT NULL,
  RES_DATE DATE NOT NULL,
  FOREIGN KEY(PNR_NUM) REFERENCES ACCOUNT(PNR),
  FOREIGN KEY(TRAIN_NUM) REFERENCES TRAIN(TRAIN_NO)
);
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Table created.

0.00 seconds

- SEAT_AVAILABILITY Table:

```
CREATE TABLE SEAT_AVAILABILITY
(
  TR_NUM INT NOT NULL,
  DATE_RES DATE NOT NULL,
  CLASS VARCHAR(20) NOT NULL,
  QUOTA VARCHAR(20) NOT NULL,
  TOTAL_SEATS INT NOT NULL,
  SEATS_AVAIL INT NOT NULL,
  FOREIGN KEY(TR_NUM) REFERENCES TRAIN(TRAIN_NO)
);
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Table created.

0.04 seconds

- ALTER TABLE:

```
ALTER TABLE TICKET ADD CLASS VARCHAR(15);
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Table altered.

0.31 seconds

DML Commands:

DML stands for Data Manipulation Language. DML commands are used for insertion, deletion and modification of data in a database. The most commonly used DML commands are:

1. INSERT: This command is used to insert new records into a relation.

```
Syntax: INSERT INTO table_name  
VALUES (value1, value2, value3  
, ...);
```

2. UPDATE: This command is used to update existing records in a relation.

```
Syntax: UPDATE table_name  
SET column1 = value1, column2 = value2, ...  
WHERE condition;
```

3. DELETE: This command is used to delete desired records from a relation.

```
Syntax: DELETE FROM table_name WHERE condition;
```

4. SELECT: This command is used to retrieve data from a database.

```
Syntax: SELECT column1, column2, ... FROM table_name;
```

Implementing DML Commands:

- INSERT

```
INSERT INTO CANCELLATION VALUES(5, 10003, 102, 'EASTERN EXPRESS', 'FIRST', 1, TO_DATE('2021-03-27', 'YYYY-MM-DD'));
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

1 row(s) inserted.

0.00 seconds

- UPDATE

```
UPDATE PASSENGERS SET OFFICE_NAME='CRAB ENTERPRISES' WHERE PERSONAL_ID=1002;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

1 row(s) updated.

0.03 seconds

- DELETE

```
DELETE FROM SEAT_AVAILABILITY WHERE TR_NUM=104;
```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

1 row(s) deleted.

0.01 seconds

- **SELECT**

```
SELECT * FROM PASSENGERS
```

PERSONAL_ID	NAME	AGE	SEX	PHONE_NUMBER	ADDRESS	OFFICE_NAME	OFFICE_ADDRESS	OFFICE_PHONE
1001	SANJAY JOSHI	35	MALE	9971823695	NEW DELHI	LOTUS ENTERPRISES	RAJENDRA NAGAR	9186293111
1002	RAGHAV SINGH	21	MALE	9821721610	HAUJ KHAS	CRAB ENTERPRISES	MOHIT NAGAR	9013283155
1003	RIYA VERMA	25	FEMALE	8129623012	GHAZIABAD	UMBRELLA CORPORATION	ROHINI	9012555936
1004	SHYAM KHANNA	40	MALE	9971682930	CHANDANI CHOWK	ROSEGOLD PRIVATE LIMITED	VASANT KUNJ	8123771954
1005	PRIYA MEHTA	22	FEMALE	9972391168	SHAHNARA	SILVERLINE INDUSTRIES	DILSHAD GARDEN	9821663189

5 rows returned in 0.00 seconds

[CSV Export](#)

CONCLUSION:

At the end of this experiment, various common DDL and DML commands were successfully implemented. This experiment afforded the opportunity to work with tables in DBMS, practically implement various constraints and brush up the basic skills of database management.

EXPERIMENT 8

AIM: Performing Simple Queries.

THEORY

SQL Queries:

Queries are used in SQL to manipulate and retrieve data from a database as per requirements. This is achieved with the help of some constraints. In SQL, the SELECT command is used for efficient query processing. The data retrieved is presented in the form a result table and the results are called result-sets.

Performing Simple Queries:

- Retrieve all records from passengers table whose age is greater than 25

☒ Autocommit

Display

10

▼

SELECT * FROM PASSENGERS WHERE AGE > 25

Results

Explain

Describe

Saved SQL

History

PERSONAL_ID	NAME	AGE	SEX	PHONE_NUMBER	ADDRESS	OFFICE_NAME	OFFICE_ADDRESS	OFFICE_PHONE
1001	SANJAY JOSHI	35	MALE	9971823695	NEW DELHI	LOTUS ENTERPRISES	RAJENDRA NAGAR	9186293111
1004	SHYAM KHANNA	40	MALE	9971682930	CHANDANI CHOWK	ROSEGOLD PRIVATE LIMITED	VASANT KUNJ	8123771954

2 rows returned in 0.07 seconds

CSV Export

- Retrieve all records from train table where duration is more than or equal to 150 minutes

☒ Autocommit

Display

10

▼

SELECT * FROM TRAIN WHERE DURATION >= 150

Results

Explain

Describe

Saved SQL

History

TRAIN_NO	TRAIN_NAME	DURATION	FIRST_STATION	LAST_STATION
102	EASTERN EXPRESS	260	RAJIV CHOWK	HAUJ KHAS
104	NORTHERN EXPRESS	180	DELHI GATE	SECTOR 28
105	NORTHEAST EXPRESS	300	NOIDA	DWARKA

3 rows returned in 0.06 seconds

CSV Export

- Display the Passenger Id, Passenger Name and Number of Seats from Reservations Table where Train_Num=101

☒ Autocommit Display 10 ▼

```
SELECT PASSENGER_ID, PASSENGER_NAME, NUMBER_OF_SEATS FROM RESERVATIONS WHERE TRAIN_NUM=101
```

Results Explain Describe Saved SQL History

PASSENGER_ID	PASSENGER_NAME	NUMBER_OF_SEATS
1001	SANJAY JOSHI	5
1002	RAGHAV SINGH	2
1005	PRIYA MEHTA	2

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

- Display those records from Account Table where Amount is between 1000 and 1500

☒ Autocommit Display 10 ▼

```
SELECT * FROM ACCOUNT WHERE AMOUNT > 1000 AND AMOUNT < 1500
```

Results Explain Describe Saved SQL History

PNR	ACCOUNT_HOLDER_NAME	DATE_OF_BOOKING	AMOUNT
10001	SANJAY JOSHI	15-MAR-21	1250
10004	SHYAM KHANNA	08-FEB-21	1050

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

- Display those records from Cancellation Table where more than one seat has been cancelled or the seat is cancelled in Eastern Express

☒ Autocommit Display 10 ▼

```
SELECT * FROM CANCELLATION WHERE NUM_OF_SEATS > 1 OR TRAIN_NAME='EASTERN EXPRESS'
```

Results Explain Describe Saved SQL History

CANCEL_NO	PNR_NUM	TRAIN_NUM	TRAIN_NAME	CLASS	NUM_OF_SEATS	RES_DATE
1	10005	105	NORTHEAST EXPRESS	FIRST	3	20-MAR-21
2	10004	103	WESTERN EXPRESS	SECOND	2	15-FEB-21
5	10003	102	EASTERN EXPRESS	FIRST	1	27-MAR-21

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

- Find out the number of trains in operation on the basis of their class

☒ Autocommit Display 10 ▼

```
SELECT CLASS, COUNT(*) FROM SEAT_AVAILABILITY GROUP BY CLASS;
```

Results Explain Describe Saved SQL History

CLASS	COUNT(*)
SECOND	1
FIRST	3
THIRD	1

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

- Calculate the total number of seats in trains based on class

☒ Autocommit Display 10 ▼

```
SELECT CLASS, SUM(TOTAL_SEATS) FROM SEAT_AVAILABILITY GROUP BY CLASS;
```

Results Explain Describe Saved SQL History

CLASS	SUM(TOTAL_SEATS)
SECOND	500
FIRST	500
THIRD	650

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

- Calculate the total number of seats in trains based on quota

☒ Autocommit Display 10 ▼

```
SELECT QUOTA, SUM(TOTAL_SEATS) FROM SEAT_AVAILABILITY GROUP BY QUOTA;
```

Results Explain Describe Saved SQL History

QUOTA	SUM(TOTAL_SEATS)
LADIES	250
GENERAL	1150
PREMIUM	250

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

- Calculate the average age of male and female passengers from the passenger table

☒ Autocommit Display 10 ▼

```
SELECT SEX, AVG(AGE) FROM PASSENGERS GROUP BY SEX;
```

Results Explain Describe Saved SQL History

SEX	AVG(AGE)
MALE	32
FEMALE	23.5

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

- Retrieve information of all bookings from Account table made between 2021-03-01 and 2021-03-20

☒ Autocommit Display 10 ▼

```
SELECT * FROM ACCOUNT WHERE DATE_OF_BOOKING BETWEEN TO_DATE('2021-03-01', 'YYYY-MM-DD') AND TO_DATE('2021-03-20', 'YYYY-MM-DD')
```

Results Explain Describe Saved SQL History

PNR	ACCOUNT_HOLDER_NAME	DATE_OF_BOOKING	AMOUNT
10001	SANJAY JOSHI	15-MAR-21	1250
10002	RAGHAV SINGH	17-MAR-21	500
10003	RIYA VERMA	10-MAR-21	200

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

CONCLUSION:

As observed in the experiment, we understood how to perform simple queries on a database. The significance of the SELECT command and the GROUP BY command was understood. Also, we understood how to use basic aggregate functions like COUNT, SUM AND AVERAGE in SQL.

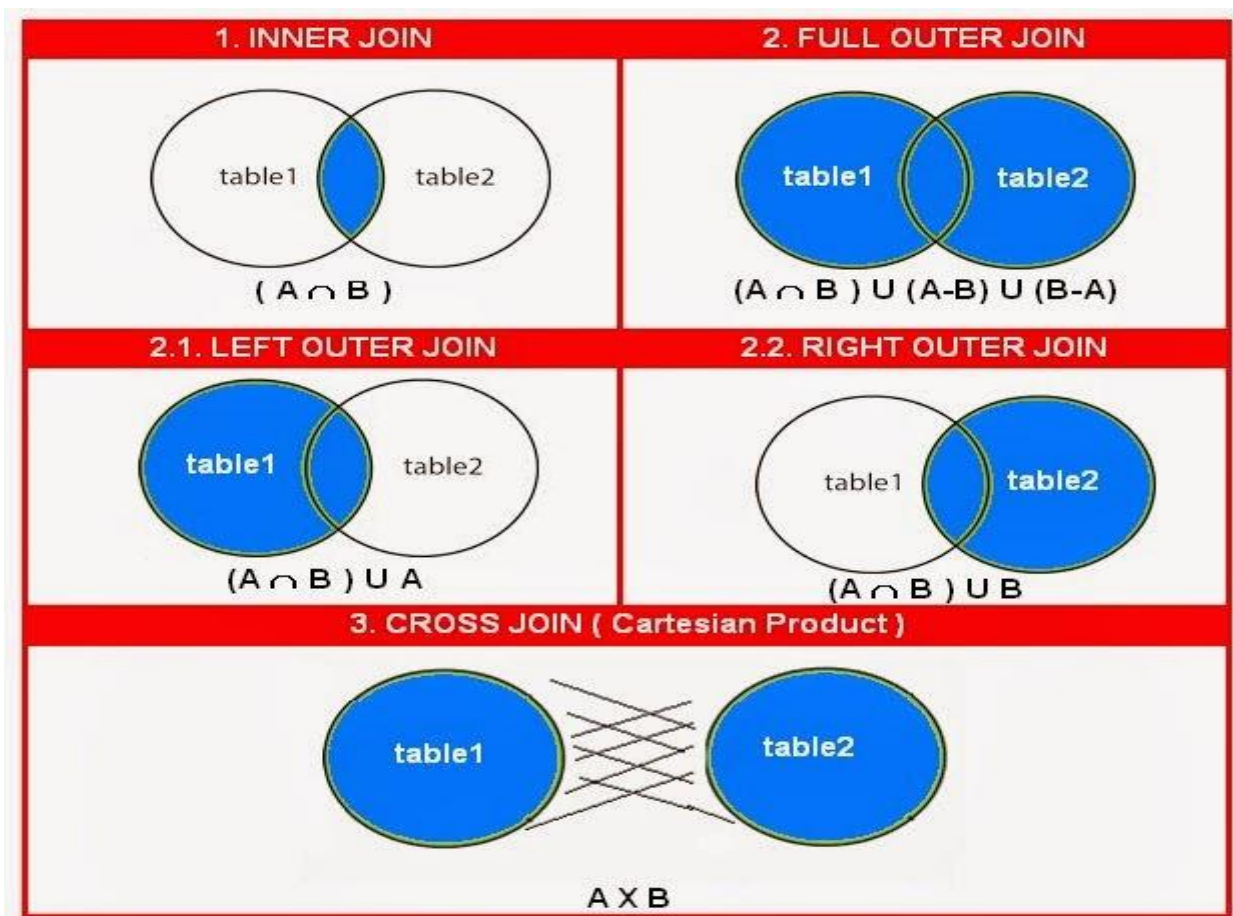
EXPERIMENT 9

AIM: Implementation of Simple Joins

THEORY

A Join is clause in SQL that is used to combine records from two or more tables based on a related attribute between them. There are five major joins in SQL. They are listed below.

- **INNER JOIN:** This join is used to retrieve records that have matching values in both relations.
- **LEFT OUTER JOIN:** This join returns all records from the left relation and matching records from the right relation.
- **RIGHT OUTER JOIN:** This join returns all records from the right relation and matching records from the left relation.
- **FULL OUTER JOIN:** This join returns all the records along with their matches in left and right relation wherever they exist.
- **CROSS JOIN:** This join returns the Cartesian product of rows from the relations in the join.



Implementation of joins:

- Inner Join

Syntax: SELECT columns FROM table1 INNER JOIN table2 ON table1.column=table2.column WHERE Condition

☒ Autocommit Display 10 ▼

```
SELECT * FROM TRAIN INNER JOIN TICKET ON TRAIN_NUMBER=TRAIN.TRAIN_NO
```

Results Explain Describe Saved SQL History

TRAIN_NO	TRAIN_NAME	DURATION	FIRST_STATION	LAST_STATION	TRAIN_NUMBER	TICKET_RATE	CLASS
101	SOUTHERN EXPRESS	120	KASHMIRI GATE	ROHINI	101	250	-
102	EASTERN EXPRESS	260	RAJIV CHOWK	HAJJ KHAS	102	200	-
103	WESTERN EXPRESS	90	AZADPUR	R.K. PURAM	103	150	-
104	NORTHERN EXPRESS	180	DELHI GATE	SECTOR 28	104	100	-
105	NORTHEAST EXPRESS	300	NOIDA	DWARKA	105	300	-

5 rows returned in 0.09 seconds [CSV Export](#)

- Left Outer Join

Syntax: SELECT columns FROM table1 LEFT OUTER JOIN table2 ON table1.column=table2.column WHERE Condition

☒ Autocommit Display 10 ▼

```
SELECT ACCOUNT.ACCOUNT_HOLDER_NAME, CANCELLATION.NUM_OF_SEATS FROM ACCOUNT LEFT OUTER JOIN CANCELLATION ON PNR_NUM=ACCOUNT.PNR
```

Results Explain Describe Saved SQL History

ACCOUNT_HOLDER_NAME	NUM_OF_SEATS
PRIYA MEHTA	3
SHYAM KHANNA	2
SHYAM KHANNA	1
SANJAY JOSHI	1
RIYA VERMA	1
RAGHAV SINGH	-

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

- Right Outer Join

Syntax: SELECT columns FROM table1 RIGHT OUTER JOIN table2 ON table1.column=table2.column WHERE Condition

☒ Autocommit Display 10

```
SELECT TRAIN.TRAIN_NAME, SEAT_AVAILABILITY.QUOTA, SEAT_AVAILABILITY.SEATS_AVAIL
FROM TRAIN RIGHT OUTER JOIN SEAT_AVAILABILITY
ON TR_NUM=TRAIN.TRAIN_NO WHERE SEAT_AVAILABILITY.SEATS_AVAIL > 50
```

Results Explain Describe Saved SQL History

TRAIN_NAME	QUOTA	SEATS_AVAIL
SOUTHERN EXPRESS	PREMIUM	100
WESTERN EXPRESS	GENERAL	150
NORTHEAST EXPRESS	LADIES	60

3 rows returned in 0.11 seconds [CSV Export](#)

- Full Outer Join

Syntax: SELECT columns FROM table1 FULL OUTER JOIN table2 ON table1.column=table2.column

☒ Autocommit Display 10

```
SELECT * FROM PASSENGERS FULL OUTER JOIN RESERVATIONS ON PASSENGER_ID=PASSENGERS.PERSONAL_ID
```

Results Explain Describe Saved SQL History

PERSONAL_ID	NAME	AGE	SEX	PHONE_NUMBER	ADDRESS	OFFICE_NAME	OFFICE_ADDRESS	OFFICE_PHONE	RESERVATION_NO	PASSENGER_ID	PASSENGER_NAME	TRAIN_NUM	NUMBER_OF_SEATS	RESERVATION_DATE
1001	SANJAY JOSHI	35	MALE	9071823885	NEW DELHI	LOTUS ENTERPRISES	RAJENDRA NAGAR	9180238311	1	1001	SANJAY JOSHI	101	5	21-MAR-21
1002	RAGHAV SINGH	21	MALE	9021721610	HALLU KHAS	CRAB ENTERPRISES	MOHT NAGAR	9013283165	2	1002	RAGHAV SINGH	101	2	21-MAR-21
1003	RIVIA VERMA	25	FEMALE	8128023012	GHAZIABAD	UMBRELLA CORPORATION	ROHINI	9012555639	3	1003	RIVIA VERMA	102	1	22-MAR-21
1004	SHYAM KHANNA	40	MALE	9071802030	CHANDANI CHOWK	ROSEGOLD PRIVATE LIMITED	VASANT KUNJ	8123771854	4	1004	SHYAM KHANNA	103	5	15-FEB-21
1004	SHYAM KHANNA	40	MALE	9071802030	CHANDANI CHOWK	ROSEGOLD PRIVATE LIMITED	VASANT KUNJ	8123771854	5	1004	SHYAM KHANNA	104	3	17-FEB-21
1005	PRIYA MEHTA	22	FEMALE	9072381188	SHAHADARA	SILVERLINE INDUSTRIES	DILSHAD GARDEN	9021683109	6	1005	PRIYA MEHTA	105	10	20-MAR-21
1005	PRIYA MEHTA	22	FEMALE	9072381188	SHAHADARA	SILVERLINE INDUSTRIES	DILSHAD GARDEN	9021683109	7	1005	PRIYA MEHTA	101	2	21-MAR-21

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

CONCLUSION:

As observed in the experiment, we understood how to perform various types of joins on relations based on a common attribute. Various types of joins were discussed and implemented with each having their own significance. Also, we understood that it is possible to combine tables even if all the records in tables do not match.

EXPERIMENT 10

AIM: Implementing Aggregate Function, Date and Time Function

THEORY

Aggregate Functions:

Aggregate functions are functions that perform a certain operation on multiple rows which are grouped together as input on certain criteria to form a single value of more significant meaning. In other words, these functions are used to extract useful information from the data which provide useful insights on the data. Most of the aggregate functions disregard NULL values apart from the COUNT function. They are used in SELECT statements usually with a GROUP BY clause. There are many aggregate functions available in SQL. Some important functions are given below.

- Count
- Avg
- Max
- Min
- Sum

COUNT:

It is used to return the number of rows that is satisfied by the queries. In other words, it is used to count the occurrences of a particular type of record in a table and return the result. It also counts NULL values.

Syntax:

```
SELECT COUNT(column_name) FROM table_name;
```

AVG:

AVG stands for average. This aggregate function is used to return the average value of all values for a particular numeric attribute ignoring the NULL values.

Syntax:

```
SELECT AVG(column_name) FROM table_name
```


MAX:

It returns the largest value among all the values for a particular attribute.

Syntax:

```
SELECT MAX(column_name) FROM table_name;
```

MIN:

It returns the smallest value among all the values for a particular attribute.

Syntax:

```
SELECT MIN(column_name) FROM table_name;
```

SUM:

It returns the sum total of all the values for a particular a particular attribute.

Syntax:

```
SELECT SUM(column_name) FROM table_name;
```

Date and Time Functions:

SQL has various functions that perform operations on date and time attributes in a table. In addition, date and time functions also help to extract the current date and time as well as examine timestamps. There are numerous date and time functions that are used to add, retrieve and manipulate date and time values. Some of them are:

- SYSDATE
- ADDDATE
- TIMESTAMP
- CURTIME
- ADDTIME
- DATE_FORMAT
- DATEDIFF
- DAYNAME
- DAYOFMONTH
- DAYOFYEAR
- DAYOFWEEK

All these functions have fairly straightforward applications

The SYSDATE function has been implemented in this experiment. Additionally, the TO_CHAR function has been used in conjunction with sysdate. Also, relation has been queries to extract records between two dates.

SYSDATE:

This particular function is used to get the current date of the system. In other words, it retrieves the date of the day this query was executed.

Syntax:

```
SELECT SYSDATE FROM DUAL
```

TO_CHAR:

It is used to convert any numeric or date data type to a string. It can be used in conjunction with SYSDATE to display current date and time.

Syntax:

```
SELECT TO_CHAR(SYSDATE, 'MM-DD-YYYY HH24:MI:SS') FROM DUAL
```

IMPLEMENTING AGGREGATE FUNCTIONS

- The average amount paid by a customer to travel inclusive of all trains

☒ Autocommit Display 10 ▼

```
SELECT AVG(AMOUNT) FROM ACCOUNT
```

Results Explain Describe Saved SQL History

AVG(AMOUNT)
1300

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

- The total number of seats available in each quota

☒ Autocommit Display 10 ▼

```
SELECT QUOTA, SUM(TOTAL_SEATS) FROM SEAT_AVAILABILITY GROUP BY QUOTA
```

Results Explain Describe Saved SQL History

QUOTA	SUM(TOTAL_SEATS)
LADIES	250
GENERAL	1150
PREMIUM	250

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

- The total number of seats remaining in each quota

☒ Autocommit Display 10 ▼

```
SELECT QUOTA, SUM(SEATS_AVAIL) AS REMAINING_SEATS FROM SEAT_AVAILABILITY GROUP BY QUOTA
```

Results Explain Describe Saved SQL History

QUOTA	REMAINING_SEATS
LADIES	100
GENERAL	170
PREMIUM	100

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

- The number of cancellations that have occurred

☒ Autocommit Display 10 ▼

```
SELECT COUNT(CANCEL_NO) AS NUMBER_OF_CANCELLATIONS FROM CANCELLATION
```

Results Explain Describe Saved SQL History

NUMBER_OF_CANCELLATIONS
5

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

- The maximum ticket fare among all the trains

☒ Autocommit Display 10 ▼

```
SELECT MAX(TICKET_RATE) AS MAX_FARE FROM TICKET
```

Results Explain Describe Saved SQL History

MAX_FARE
300

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

- The minimum ticket fare among all trains

☒ Autocommit Display 10 ▼

```
SELECT MIN(TICKET_RATE) AS MIN_FARE FROM TICKET
```

Results Explain Describe Saved SQL History

MIN_FARE
100

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

- The longest time taken by a train

☒ Autocommit Display 10 ▼

SELECT MAX(DURATION) AS LONGEST_PATH_TAKE FROM TRAIN

Results Explain Describe Saved SQL History

LONGEST_PATH_TAKE
300

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

- The shortest time taken by a train

☒ Autocommit Display 10 ▼

SELECT MIN(DURATION) AS SHORTEST_PATH_TAKES FROM TRAIN

Results Explain Describe Saved SQL History

SHORTEST_PATH_TAKES
90

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

IMPLEMENTING DATE AND TIME FUNCTIONS

- Extract current date of the system

☒ Autocommit Display 10 ▼

```
SELECT SYSDATE FROM DUAL
```

Results Explain Describe Saved SQL History

SYSDATE
27-APR-21

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

- Extract system sate and time as a string

☒ Autocommit Display 10 ▼

```
SELECT TO_CHAR(SYSDATE, 'Dy DD-MON-YYYY HH24:MI:SS') AS CURRENT_TIME FROM DUAL
```

Results Explain Describe Saved SQL History

CURRENT_TIME
Tue 27-APR-2021 21:51:32

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

- Retrieve records of all the bookings made between March 20, 2021 and March 22, 2021

☒ Autocommit Display 10 ▼

```
SELECT * FROM RESERVATIONS WHERE RESERVATION_DATE BETWEEN '20-MAR-21' AND '22-MAR-21'
```

Results Explain Describe Saved SQL History

RESERVATION_NO	PASSENGER_ID	PASSENGER_NAME	TRAIN_NUM	NUMBER_OF_SEATS	RESERVATION_DATE
1	1001	SANJAY JOSHI	101	5	21-MAR-21
2	1002	RAGHAV SINGH	101	2	21-MAR-21
3	1003	RIYA VERMA	102	1	22-MAR-21
6	1005	PRIYA MEHTA	105	10	20-MAR-21
7	1005	PRIYA MEHTA	101	2	21-MAR-21

5 rows returned in 0.03 seconds

[CSV Export](#)

CONCLUSION:

In this experiment, we were able to apply aggregate functions to various relations in the database which helped in better retrieval of valuable information about the data. Additionally, we also implemented date and time functions which helped us work with date and time attributes.

EXPERIMENT 11

AIM: Implementing Triggers, Views, Grant and Revoke commands

THEORY

TRIGGERS:

Triggers are stored procedures that automatically invokes whenever a particular event occurs in the database. They can be used along with DML commands such as INSERT, UPDATE AND DELETE whenever a user tries to modify records using these commands.

Classification of triggers:

Triggers are classified into various types on various grounds. They can be classified as follows.

- On the basis of timing
 - **BEFORE Trigger:** These triggers are meant to execute before a particular event occurs. For example: before inserting a particular record in a table. These triggers allow for write operations to occur in a database.
 - **AFTER Trigger:** These triggers are meant to execute after a particular event occurs. For example: Copying new records to a separate table after records are inserted in a table. These triggers allow only for read operations.
 - **INSTEAD OF Trigger:** These are special triggers that fires before the SQL Server starts the execution of the action that fired it. They can be used in conjunction with DML commands such as INSERT, UPDATE and DELETE.
- On the basis of level
 - **ROW Level Trigger:** These triggers fire each time a row is inserted in a relation that satisfies the given criteria.
 - **STATEMENT or COLUMN Level Triggers:** These triggers execute only once for a specified event statement and affects the database at a column level.
- On the basis of event
 - **DML Triggers:** These triggers are executed when DML events are specified such as INSERT, DELETE or UPDATE.
 - **DDL Triggers:** These triggers are executed when DDL events are specified such as CREATE or ALTER.
 - **DATABASE Triggers:** These triggers are executed when DATABASE events are specified such as LOGON, LOGOFF, STARTUP or SHUTDOWN.

Syntax:

```
CREATE TRIGGER [trigger_name]
[BEFORE | AFTER]
{INSERT | UPDATE | DELETE}
ON [table_name]
[FOR EACH ROW]
```

Explanation of syntax:

1. **create trigger [trigger_name]:** Creates or replaces an existing trigger with the **trigger_name**.
2. **[before | after]:** This specifies when the trigger will be executed.
3. **{insert | update | delete}:** This specifies the DML operation.
4. **on [table_name]:** This specifies the name of the table associated with the trigger.
5. **[for each row]:** This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.
6. **[trigger_body]:** This provides the operation to be performed as trigger is fired

VIEWS:

Views are virtual tables that contain a selective portion of the data from one or more tables. It contains records and attributes like a real relation but do not contain data of their own. It only displays data of other relations. It is used as a means to restrict access to the database and also to hide the complexity of the database. A view can be created just like a normal relation using a **CREATE** statement and records can be selected using the **SELECT** statement. DML operations can also be performed on a view but it also affects the data in the original relation.

Syntax:

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```

Explanation on syntax:

- **view_name:** Name for the View
- **table_name:** Name of the table

- **condition:** Condition to select rows

GRANT AND REVOKE COMMANDS:

Grant commands are types of DCL commands that are used to grant or restrict access to a database. These commands are only available to database administrator and are used to monitor how much privilege a user holds when working with a database. Some users can only insert records while the others can only view them. Grant and revoke commands also allow for temporary access to a user as the privileges can be revoked from them when the job is done.

- **Grant:** SQL Grant commands are used to grant privileges to other users in a database. The database administrator may grant privileges to certain users or all users at once as per the requirement. There is also a provision that allows the database administrator to allow users to grant privileges to other users using GRANT OPTION.

Syntax:

```
GRANT privilege_name on object_name  
TO {user_name | public | role_name}  
[WITH GRANT OPTION]
```

- **Revoke:** SQL Revoke commands are used to take back or restrict privileges from a user after their job is done and the said access is no longer necessary to them. This can only be done by a database administrator. It is also possible to revoke selective privileges from them.

Syntax:

```
REVOKE privilege_name on object_name  
FROM {user_name | public | role_name}
```

IMPLEMENTATION OF TRIGGERS:

- Create a trigger DISCOUNT

☒ Autocommit Display 10 ▼

```
CREATE OR REPLACE TRIGGER DISCOUNT
BEFORE INSERT
ON ACCOUNT
FOR EACH ROW
BEGIN
IF (:NEW.AMOUNT >= 2000)
THEN :NEW.AMOUNT := :NEW.AMOUNT - :NEW.AMOUNT*0.1;
END IF;
END;
```

Results Explain Describe Saved SQL History

Trigger created.

0.06 seconds

- Insert a new record in ACCOUNT

☒ Autocommit Display 10 ▼

```
INSERT INTO ACCOUNT VALUES(10006, 'SHAHID KHAN', TO_DATE('2021-02-15', 'YYYY-MM-DD'), 2550 )
```

Results Explain Describe Saved SQL History

1 row(s) inserted.

0.00 seconds

- Check if a 10% discount has been provided to PNR='10006'

☒ Autocommit Display 10 ▼

```
SELECT * FROM ACCOUNT WHERE PNR=10006
```

PNR	ACCOUNT_HOLDER_NAME	DATE_OF_BOOKING	AMOUNT
10006	SHAHID KHAN	15-FEB-21	2295

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

A 10% discount has been provided on the total amount of 2550. The new amount is 2295.

IMPLEMENTATION OF VIEWS:

- **Create a View BEST_TRAIN**

☒ Autocommit Display 10 ▼

CREATE VIEW BEST_TRAIN AS
SELECT * FROM SEAT_AVAILABILITY WHERE CLASS='FIRST'

Results Explain Describe Saved SQL History

View created.

0.00 seconds

- **View BEST_TRAIN**

☒ Autocommit Display 10 ▼

SELECT * FROM BEST_TRAIN

Results Explain Describe Saved SQL History

TR_NUM	DATE_RES	CLASS	QUOTA	TOTAL_SEATS	SEATS_AVAIL
101	21-MAR-21	FIRST	PREMIUM	250	100
102	27-MAR-21	FIRST	LADIES	150	40
105	20-MAR-21	FIRST	LADIES	100	60

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

IMPLEMENTATION OF GRANT AND REVOKE COMMANDS

- Create a new user “user1”

☒ Autocommit Display ▼

CREATE USER user1 IDENTIFIED BY userpass

Results Explain Describe Saved SQL History

User created.

0.06 seconds

- Grant Select, Update, Insert and Delete Privileges To user1

☒ Autocommit Display ▼

GRANT SELECT, INSERT, UPDATE, DELETE ON PASSENGERS TO user1

Results Explain Describe Saved SQL History

Statement processed.

0.02 seconds

- Grant the GRANT OPTION to user1

☒ Autocommit Display ▼


GRANT INSERT(TRAIN_NAME), UPDATE, DELETE ON TRAIN TO user1 WITH GRANT OPTION;

Results Explain Describe Saved SQL History

Statement processed.

0.00 seconds

- **Revoke the update privilege from user1**

☒ Autocommit Display 10 
REVOKE UPDATE ON TRAIN FROM user1

Results Explain Describe Saved SQL History

Statement processed.

0.01 seconds

CONCLUSION:

In this experiment, we learnt how the significance of triggers and also implemented them. The impact of the trigger was also observed by inserting a record that would cause the trigger to initiate. Additionally, we also implemented views and observed the selections made based on the criteria provided. Finally, grant and revoke commands were also implemented. For this, we first created a user and granted them certain privileges and subsequently revoked them after the job was done. All of them were successfully built and executed.