4.1 WHY NORMALIZATION (INSERTION, UPDATING, DELETION ANOMALIES)

Normalization:

Data normalization is a process of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like insertion, updation and deletion.

It is a multi-step process that puts data into tabular form by removing duplicated data from the relation table.

Normalization is used for mainly two reasons:

- There is no redundancy of data (all data is stored in only one place).
- Data dependencies are logical (all related data items are stored together).

Normalization involves refactoring a table into smaller (and less redundant) tables but without losing information; defining foreign keys in the old table referencing the primary keys of the new ones.

The objective is to isolate data so that additions, deletions, and modifications of an attribute can be made in just one table and then propagated through the rest of the database using the defined foreign keys.

Problem without Normalization:

Without normalization it become difficult to handle and update the database without facing data loss.

	Student		
S_id	**S_name**	**S_add**	**Subject**
401	Paresh	Surat	Bio
402	Suresh	Baroda	Maths
403	Ramesh	Ahmedabad	Maths
404	Paresh	Surat	Physics
405			

- **Insertion Anomaly:**

  Suppose For a new admission, we have a stud_id, name and address but still student has not opted for any subject yet and subject column does not allow null values then leading to insert anomaly.

- **Deletion Anomaly:**

  If student 402 has only one subject and temporary he drops it, when we delete that row, entire student record will be deleted along with it.

- **Update Anomaly:**

  To update address of student Paresh who occurs twice or more than twice in a table, we will have to update s_add column in all rows, else data will become inconsistency.

Update, delete and insert anomalies are very undesirable in any database. So these anomalies are avoided by the process of normalization.

Need for normalization:

- Improves database design.
- Ensure minimum redundancy of data.
- It can save storage space and ensure the consistency of your data.
- More flexible database structure.
- Removes anomalies for database activities.

4.2 NORMALIZATION RULES:

4.2.1 Concepts of Dependency, Transitive Dependency

The attributes of a table is said to be dependent on each other when an attribute of a table uniquely identifies another attribute of the same table.

A functional dependency is an association between two attributes of the same relational database table. One of the attributes is called the determinant and the other attribute is called the determined. For each value of the determinant there is associated one and only one value of the determined.

A functional Dependency is a constraint between two attributes in a table from a database that means functional dependency is a constraint that describe the relation between attributes in a relation. This can be written as:

A -> B

That means B is functionally dependent on A.

An attribute in a relational model is said to be a functional dependent on another attribute in a table. If it can take only one value for a given value of a attribute upon which it is functionally dependent.

Note:

When non-key attributes are dependent on-key attribute then we can say that non-key attributes are functionally dependent on-key attributes.

For given relation R, attributes A of R is functionally dependent on attribute B of R, if and only if every value of A of R is dependent on every value of B of R.

Example:

Supplier			
S_NO	**S_Name**	**Status**	**City**
S1	Ankit	20	Baroda
S2	Ankit	10	Baroda
S3	Ankit	10	Baroda

Here, S_Name is functionally dependent on S_No because S_Name can take only one value for the given value of S_No. so that,

S_No -> S_Name

Similarly, City and Satus is functionally depedent on S_No because for each of S_No there will be the one City and Status.

S_No -> City

S_No ->Status

S_No -> (S_Name., City, Status)

Part				
P_NO	**P_Name**	**Color**	**Weight**	**City**
P1	Nut	Red	12	Baroda
P2	Nut	Green	17	Surat
P3	Nut	Red	17	Surat

Shipment		
S_No	**P_NO**	**Qty**
S1	P1	270
S1	P2	270
S1	P3	430
S2	P1	430
S2	P3	500

Here for shipment table S_No and P_NO tends to Qty because each combination of S_No and P_No result only 1 Qty.

4.2.2 Armstrong Axioms

Armstrong's Axioms **OR** **Inference Rule for Functional Dependency:** **Consider following Table:** **Student(Rno, Name, Address, Pincode)**	
1) Reflexivity Rule:	**If Y is a Subset of X then X -> Y** **Example:** If {Rno,Name} -> {Name} So we can say that Functional dependency {Rno,Name} -> {Name} Holds.

2) Augmentation Rule:	**If X -> Y then {X,Z} -> {Y,Z} OR {Z,X} -> {Z,Y}.** **Example**: If {Rno} -> {Address} Then {Rno, Name} -> {Address, Name} OR {Name, Rno} -> {Name, Address}.
3) Transitivity Rule:	**If X->Y and Y->Z Then X->Z.** **Example:** If {Rno} -> {Address} and {Address} -> {Pincode} then transitivity {Rno} -> {Pincode}. Here, Address is FD on Rno and Pincode is FD on Address. So that Pincode is FD on Rno.
4) Decompositio n Rule OR Projectivity Rule:	**If X -> {Y, Z} then X->Y and X->Z.** **Example:** If {Rno} -> {Name, Address} then {Rno} -> {Name} and {Rno} -> {Address}. Here, Name and Address are functionally dependent on Rno so we can say that functional Dependencies {Rno} -> {Name} and {Rno} -> {Address} holds.

5) Union Rule:	**If X ->Y and X->Z then X -> {Y, Z}.** **Example:** If {Rno} -> {Name} and {Rno} -> {Address} Then {Rno} -> {Name,Address}.
6) Composition Rule:	**If X -> Y and Z -> W then {X,Z} -> {Y,W}** **Example:** {Rno} -> {Name} and {Address} -> {Pincode} Then {Rno, Address} -> {Name, Pincode}.
7) Pseudo Transitivity Rule:	**If X -> Y and {Y,W} -> Z then {X,W} -> Z.** **Example:** Rno -> Name and {Name,Address} -> {Pincode} then {Rno, Address} -> {Pincode}. Here, Name is functionally depends on Rno and pincode is functionally Depends on Name and Address so we can say that the functional dependency {Rno, Address} -> {Pincode} holds.

4.2.3 1st Normal Form, 2nd Normal Form, 3rd Normal Form, B.C.N.F.

Normalization Rule:

- **First Normal Form (1NF)**
- **Second Normal Form (2NF)**
- **Third Normal Form (3NF)**
- **Boyce-Codd Normal Form (BCNF)**

First Normal Form(1NF):

As per the 1NF, no two rows of data must contain repeating group of information. Each set of column must have a unique value, such that multiple columns cannot be used to fetch the same row.

First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains.

Each table should be organized into row, and each row should have primary key that distinguishes it as unique.

Student_Name	Age	Subject	Contact_No
Aakash	20	Bio	9825760188 8460770242
Paresh	15	Maths	8745637241
Suresh	16	Maths	9923456789
Ramesh	17	Bio	8723456790

In 1NF, any column must not have more than one value. So we must separate such data into multiple rows.

This table is **not in 1NF** as the rule says "each attribute of a table must have atomic (single) values", the Contact_No values for student Aakash violates that rule.

To make table in 1NF we should have the data like:

Student_Name	Age	Subject	Contact_No
Aakash	20	Bio	9825760188
Aakash	20	Bio	8460770242
Paresh	15	Maths	8745637241

Suresh	16	Maths	9923456789
Ramesh	17	Bio	8723456790

Using 1NF, there will be many columns with same value in multiple row but each row as whole will be unique.

Second Normal Form(2NF):

Firstly, what is prime and non-prime attribute?

Prime-attribute:

An attribute which is a part of the candidate-key, is known as prime attribute.

Non-Prime attribute:

An attribute, which is not a part of the prime-key, is said to be non-prime attribute.

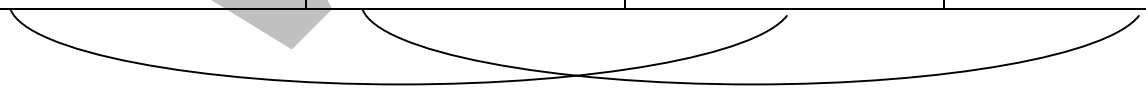
If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.

Table must be in 1NF.

As per 2NF, there must not be any partial dependency of any column on Primary key. Each column in the table that is not part of the primary key must depend upon the entire candidate key for its existence.

Example:

Student_project			
Stud_ID	**Proj_ID**	**Stud_Name**	**Proj_Name**



In this Student_project table the prime ley attributes are Stud_ID and Proj_ID. According to the rule, non-key attributes, i.e. Stu_Name and Proj_Name must be dependent upon both and not on any of the prime key attribute individually.

But we find that Stu_Name can be identified by Stu_ID and Proj_Name can be identified by Proj_ID independently.

This is called **partial dependency**, which is not allowed in Second Normal Form.

Example:

Student		
Stud_ID	**Stud_Name**	**Proj_ID**

Project	
Proj_ID	**Proj_Name**

We broke the table into two tables. So there exists no partial dependency. Now both above table qualifies for second NF and never suffer from update anomalies.

Third Normal Form(3NF):

A table is said in 3NF if both following condition hold:

- Table must be in 2NF.
- **Transitive functional dependency** of non-prime attribute on any other non-prime attribute should be removed.

  Transitive functional dependency:

  A transitive dependency in a database is an indirect relationship between values in the same table that causes a functional dependency.

  A transitive dependency requires three or more attributes (or database columns) that have a functional dependency between them, meaning that Column A in a table relies on Column B through an intermediate Column C.

3NF applies that every non-prime attribute of table must be dependent on primary key or we can say that there should not be the case that non-prime attribute is determine by another nonprime attribute. So that transitive function dependency should be removed from the table and also the table must be in 2NF.

Example:

Student					
Stud_id	**Name**	**DOB**	**City**	**State**	**Zip**

Stud_id is a primary key of a table but City and State depends on Zip.

Stud_id -> {City, State}

And

{City, State} -> Zip

Then

Stud_id -> Zip.

Should be removed from a table.

Dependency between Zip and other fields called transitive dependency.

To apply 3NF, we need to move City and State to new table with Zip Primary key.

Student			
Stud_id	**Name**	**DOB**	**Zip**

Zipcodes		
Zip	**City**	**State**

Advantages:

- Amount of data duplication reduced.
- Achieve data integrity.

Boyce-Codd Normal Form (BCNF):

Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms. A relation is in BCNF if and only if every determinant is a super key. BCNF states that −

- For any non-trivial functional dependency, X → A, X must be a super-key.

In the above table, Stud_Id is the super-key in the relation Student and Zip is the super-key in the relation ZipCodes. So,

Stu_ID → Name, DOB, Zip

And

Zip → City, State

Which confirms that both the relations are in BCNF.

4.3 CONCEPTS OF STRUCTURE QUERY LANGUAGE (SQL)

Introduction to SQL:-

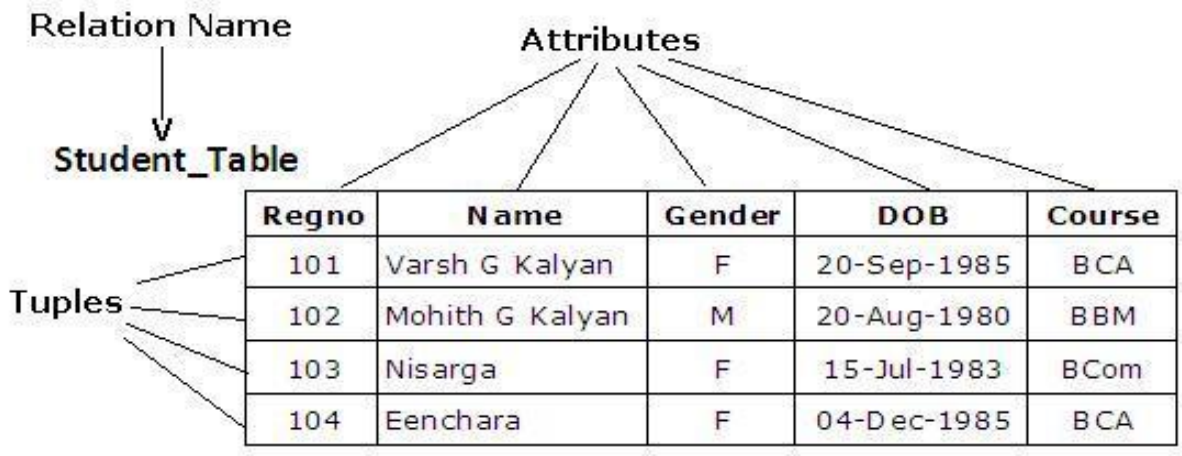


Fig: The attributes and tuples of a relation STUDENT_Table

Query Language:

- **SQL** stands for Structured Query Language. SQL lets you access and manipulate databases.
- **Query Language:**
  Query language is a high level language in which user requires information from the database by writing a query which is then evaluated by query processor. Query language can be categorized into either **procedural** or **non-procedural.**

  - **Procedural:**

  In procedural language the user instructs the system to perform a series of operations on the database to compute desire result.
  It required user to specify which data are needed and **how to get those data.**

  - **Non-Procedural:**

  In non-procedural language the user describe desire information **without giving a specified procedure** for obtaining the information.

4.3.1 SQL datatypes : int, float, double, number, char, varchar, varchar2, Text, date

DATA TYPE	DESCRIPTION
INT(*size*)/ INTEGER(*size*)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The *size* parameter specifies the maximum display width (which is 255)
FLOAT(*size*, *d*)	A floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
FLOAT(*p*)	A floating point number. MySQL uses the *p* value to determine whether to use FLOAT or DOUBLE for the resulting data type. If *p* is from 0 to 24, the data type

	becomes FLOAT(). If *p* is from 25 to 53, the data type becomes DOUBLE()
DOUBLE(*size*, *d*)	A normal-size floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter
numeric(p,s)	▪ Fixed precision and scale numbers. ▪ Allows numbers from -10^38 +1 to 10^38 –1. ▪ The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. ▪ The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The *size* parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The *size* parameter specifies the maximum column length in characters - can be from 0 to 65535
VARCHAR2(size)	To store variable-length character strings, you use the Oracle **VARCHAR2** data type. A **VARCHAR2** column can store a value that ranges from 1 to 4000 bytes. It means that for a single-byte character set, you can store up to 4000 characters in a **VARCHAR2** column.
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'

Difference Between Varchar and Varchar2

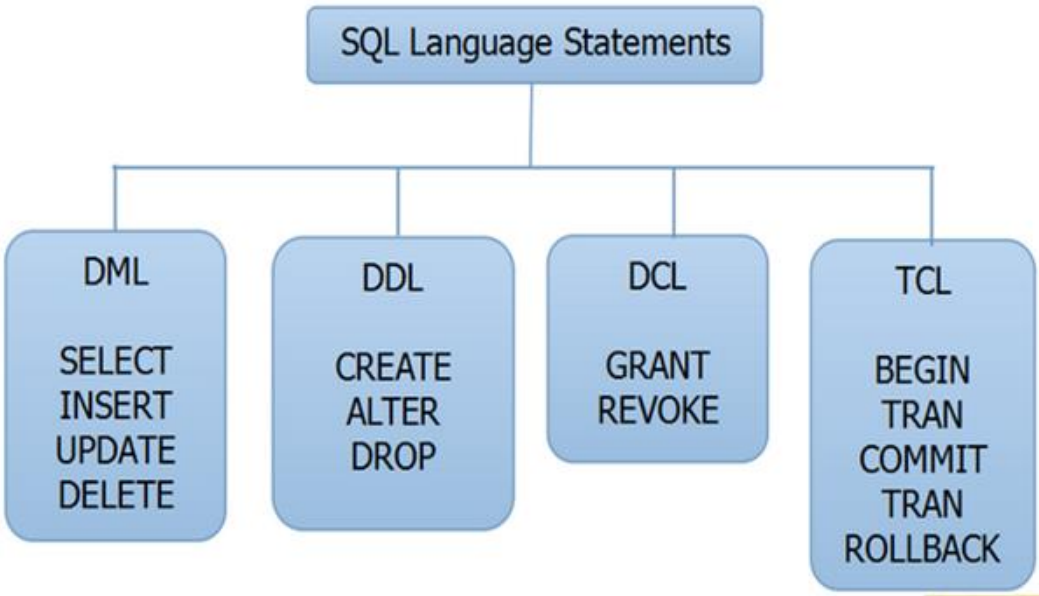
Varchar	Varchar2
Varchar can identify NULL and empty string separately.	Varchar2 cannot identify both separately. Both considered as same for this.
Varchar can store minimum 1 and maximum 2000 bytes of character data.	Varchar2 can store minimum 1 and maximum 4000 bytes of character data.Varchar2 can store minimum 1 and maximum 4000 bytes of character data.
Allocate fixed size of data irrespective of the input.	Allocate variable size of data based on input.
Allocate fixed size of data irrespective of the input. Ex: We defined varchar (15) and entered only 10 characters. But it allocates space for entire 15 characters.	Allocate variable size of data based on input. Ex: We defined varchar2 (15) and entered only 10 characters. Then varchar2 will allocate space for 10 characters only but not for 15.
For varchar data, extra spaces are padded to the right side.	For varchar2 extra spaces will be truncated.
Varchar is ANSI Sql standard	Varchar2 is Oracle standard
Varchar definition may change in future.	Varchar2 definition will not change. It is standard.
Varchar is an external datatype.	Varchar2 is an internal datatype.

SQL STATEMENTS:-

SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that **use SQL** are: Oracle, Sybase, Microsoft **SQL** Server, Access, Ingres, etc.

SQL statements are categorized into four different types of statements, which are

1. DML (DATA MANIPULATION LANGUAGE)
2. DDL (DATA DEFINITION LANGUAGE)
3. DCL (DATA CONTROL LANGUAGE)
4. TCL (TRANSACTION CONTROL LANGUAGE)



4.4 DDL STATEMENTS :

- Data Definition Language is used to define the database structure or table. DDL is also used to specify additional properties of the data.
- The storage structure and access methods used by the database system by a set of statements in a special type of DDL called a data storage and definition language.
- These statements define the implementation details of the database schema, which are usually hidden from the users. The data values stored in the database must satisfy certain consistency constraints.
- For example, suppose the university requires that the account balance of a department must never be negative. The DDL provides facilities to specify such constraints. The database system checks these constraints every time the database is updated.

4.4.1 Create , Drop, Truncate, Rename, Alter

Statement	Description
CREATE	Create new database/table.
SYNTAX	CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);
EXAMPLE	Create database university; create table department (dept_name char(20), building char(15),

	budget numeric(12,2));
ALTER OR RENAME	Modifies/Rename the structure of database/table.
SYNTAX	**To add column in a table:** Alter table table_name Add column_name datatype(size); **To delete column from table:** Alter table table_name drop column column_name; **To change the datatype or size of a column in a table:** Alter table table_name Alter column column_name datatype(size);
EXAMPLE	**Add column:** alter table student add Contact_no Numeric(10); **Delete column:** alter table student drop column contact_no; **Alter datatype of column:** alter table student alter column contact_no numeric(10);
DROP	Deletes a database/table.
SYNTAX	DROP TABLE TABLE_NAME; OR DROP DATABASE DATABASE_NAME;
EXAMPLE	Drop database university; Drop table student;
TRUNCATE	Remove all table records including allocated table spaces. TRUNCATE operation is used to delete all table records.
SYNTAX	TRUNCATE TABLE table_name;
EXAMPLE	TRUNCATE TABLE student;

4.5 DML AND DQL STATEMENTS :

- **DML(DATA MANIPULATION LANGUAGE):**

DML statements are used for managing data with in schema objects.
DML are of two types –

1. **Procedural DMLs** : require a user to specify what data are needed and how to get those data.
2. **Declerative DMLs** (also referred as **Non-procedural DMLs**) : require a user to specify what data are needed without specifying how to get those data.

   Declarative DMLs are usually easier to learn and use than procedural DMLs. However, since a user does not have to specify how to get the

data, the database system has to figure out an efficient means of accessing data.

4.5.1 Insert, Update, Delete

Statement	Description
INSERT	INSERT command is used to add new rows into the database table.
SYNTAX	**Insert data into specific columns:** Insert into table_name(Column_name1, Column_name2, Column_name3) VALUES (value1, valu2, value3); **Insert data into all columns:** Insert into table_name VALUES (value1, valu2, value3);
EXAMPLE	**Insert data into all columns:** insert into student VALUES (1,'Mohini','BCA',250,8460770222); Insert data into specific column: insert into student(roll_no,name,stream) values(2,'Patel','BBA');
UPDATE	UPDATE statement modifies records into the existing table.
SYNTAX	UPDATE *table_name* SET *column1 = value1, column2 = value2, ...* WHERE *condition*; **Note**: Here **where** clause specifies which records that you want to update. If there is no where clause, it will be updated all records.
EXAMPLE	update student set contact_no=7896541136 where roll_no=2; **Update Statement without Where clause:** update student set contact_no=8460770242; (it will replace all mobile numbers with given number in example)
DELETE	deletes all records from a table, space for the records remain
SYNTAX	DELETE FROM *table_name* WHERE *condition*;
EXAMPLE	delete from student where roll_no=6; **OR** delete from student where name='Riya' AND stream='BCA';
SELECT	SELECT query is used to retrieve a data from SQL tables.
SYNTAX	**To get specific columns from the table:** Select column1_name, column2_name From table_name;

	To get all columns from the table: Select * from table_name;
EXAMPLE	**To get specific columns from the table:** Select roll_no,name From STUDENT; **To get all columns from the table:** Select * from STUDENT;

❖ **DQL(DATA QUERY LANGUAGE):**

DQL (Data Query Language) :
DML statements are used for performing queries on the data within schema objects. The purpose of DQL Command is to get some schema relation based on the query passed to it.

4.5.2 select

Example of DQL:

- **SELECT** – is used to retrieve data from the a database.

Statement	Description
SELECT	SELECT query is used to retrieve a data from SQL tables.
SYNTAX	**To get specific columns from the table:** Select column1_name, column2_name From table_name; **To get all columns from the table:** Select * from table_name;
EXAMPLE	**To get specific columns from the table:** Select roll_no,name From STUDENT; **To get all columns from the table:** Select * from STUDENT;
SELECT WITH WHERE CLAUSE	It is use to get specific columns from the table
SYNTAX	**To get specific columns from the table:** Select column1_name, column2_name From table_name Where column_name=value;
EXAMPLE	**To get specific columns from the table:** Select ROLL_NO,NAME From STUDENT Where STREAM='BBA';

❖ DCL (DATA CONTROL LANGUAGE):

A DCL is used to control access to data in a database. It is used to give or take permission or rights on objects.

- **Grant:** A privilege can be granted to a user with a help of grant statement. A privilege assigned can be select, alter, delete, insert etc.
- **Revoke:** It is used to cancel previously granted permission.

❖ TCL (TRANSACTION CONTROL LANGUAGE):

TCL statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transaction. **Transaction** is a unit of program in execution that access and possibly update various data item.

- **Commit:** Save work done.
- **Savepoint**: Identify a point in a transaction to which you can later rollback.
- **Rollback:** Restore database to original since the last commit.

❖ DIFFERENCE BETWEEN DROP, DELETE AND TRUNCATE:-

S.NO	DROP	TRUNCATE
1.	The DROP command is used to remove table definition and its contents.	Whereas the TRUNCATE command is used to delete all the rows from the table.
2.	In the DROP command, table space is freed from memory.	While the TRUNCATE command does not free the table space from memory.
3.	DROP is a DDL(Data Definition Language) command.	Whereas the TRUNCATE is also a DDL(Data Definition Language) command.

4.	In the DROP command, view of table does not exist.	While in this command, view of table exist.
5.	In the DROP command, integrity constraints will be removed.	While in this command, integrity constraints will not be removed.
6.	In the DROP command, undo space is not used.	While in this command, undo space is used but less than DELETE.
7.	The DROP command is quick to perform but gives rise to complications.	While this command is faster than DROP.
S.NO	**DELETE**	**TRUNCATE**
1.	The DELETE command is used to delete specified rows(one or more).	While this command is used to delete all the rows from a table.
2.	It is a DML(Data Manipulation Language) command.	While it is a DDL(Data Definition Language) command.

3.	There may be WHERE clause in DELETE command in order to filter the records.	While there may not be WHERE clause in TRUNCATE command.
4.	In the DELETE command, a tuple is locked before removing it.	While in this command, data page is locked before removing the table data.
5.	We can rollback the data even after using DELETE command.	While in this command, we can't rollback.
6.	DELETE command is slower than TRUNCATE command.	While TRUNCATE command is faster than DELETE command.