Assignment Title: Introduction to Database

This Assignment is Submitted By "Group-5 (Hackers)"

Section: C

Grou p Info	
ID	**Name**
19-39902-1	**Abdur Rahman**
19-39862-1	**Fatin Ishrak Rahman**
19-40620-1	**Sadia Haque Rupa**
19-40582-1	**Tasmia Islam**
19-39815-1	**Jeba Fawjia**
19-39751-1	**Tasnim Bintey Khairul**
18-37608-1	**Ilma Nigar**

Introduction to Database

What is Database:

Database is nothing but it's a collection of data. It's a set of programs to store, access and retrieve those data in an easy and efficient way. Data within the most common types of database in operation today is typically modeled in rows and columns in a series of tables to make processing and data querying efficient. Most databases use structured query language (sql) for writing and querying data.

Database Management System:

Database management system (DBMS) is a software which is used to manage database. A database is usually controlled by a database management system. It's a collection of inter related data. There are many types of database management system software. Oracle 9i/10g/11g/xe , my sql and many other Platform where we can create database.

Application of Database:

Database touch all aspects of our lives. Banking, Airlines, Universities, Sales, Manufacturing, Human resources all places we can see the use of database.

Drawbacks of using file system for data storage:

Previously data is stored in files.it has many problems. Such as

1.**Data redundancy and inconsistency**:
Data redundancy is defined as the storing of the same data in multiple locations. Data inconsistency is when the same data exists in different formats in multiple tables.

2.**Difficulty in accessing data.**
3.**Data isolation:**

Data isolation is a property that determines when and how changes my operation and data become visible to other concurrent users and systems.

4.Data security

5.Transaction problem

6.Integrity problem

7.**View of data:**

Major purpose of a database system is to provide users with an abstract view of the data So,

Database system offer solution to all these problem.

Levels of Abstraction:

Many database system users are not computer trained, developers, hide the complexity from users through several levels of abstraction to simplify users interaction with the system.

Physical Level: Lowest level, describes how a record is stored, complex low-level data structures.

Logical Level: Next higher level, what data are stored in database and the relationships among the data.

Instances and Schemas:

Instances: What type of data and what kind of data is contain in my database then its called Instances.

Schema: What will be contained in my database then its called Schema. There are two types of Schema

1.Logical Schema

2.Physical Schema

Relational Model:

The relational model uses a collection of tables to represent both data and the relationship among both table.

E-R Data Model:

The E-R data model is based on a perception of a real world that consist of a collection of basic objects called entities and of relationships among these objects used for database design. Models an enterprise as a collection of entities and relationships.

Entities

An entity is an object that exists. It doesn't have to do anything; it just has to exist. In database administration, an **entity** can be a single thing, person, place, or object. Data can be stored about such entities.
In database administration, only those things about which data will be captured or stored is considered an entity. If you aren't going to capture data about something, there's no point in creating an entity in a database.

If you're creating a database of your employees, **examples of entities** you may have include employees and health plan enrollment.

Attribute

In general, an attribute is a characteristic. In a database management system (DBMS), an attribute refers to a database component, such as a table.

It also may refer to a database field. Attributes describe the instances in the column of a database.

For example, Roll_No, Name, DOB, Age, Address, Mobile_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.

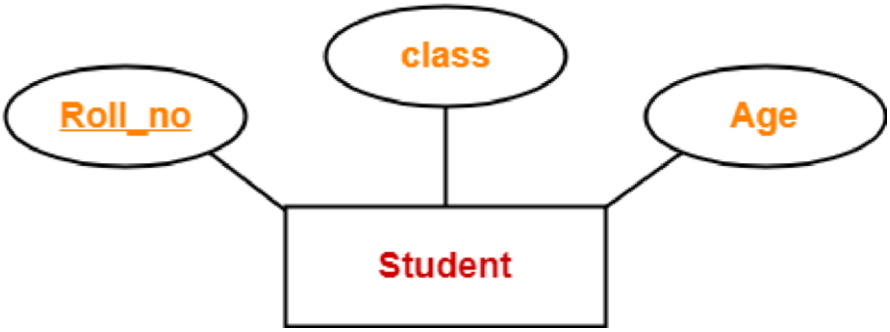
Attribute types:

- **Simple attributes;**
- **composite attributes;**
- **Single-valued attributes; multivalued attributes; Derived attributes.**

Simple Attributes-

Simple attributes are those attributes which can not be divided further.

Example-

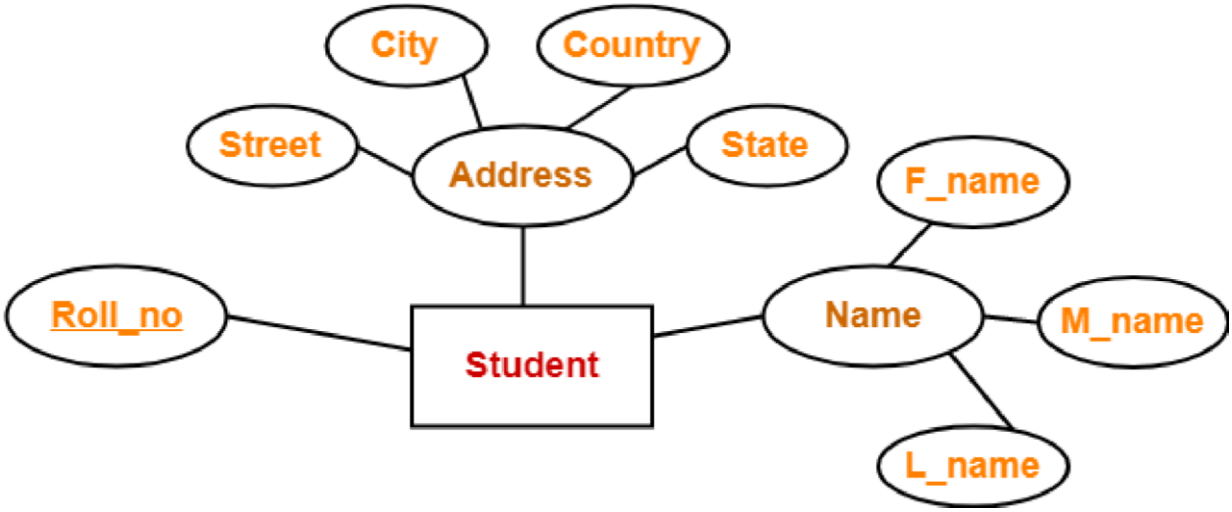


Here, all the attributes are simple attributes as they can not be divided further.

Composite Attributes-

Composite attributes are those attributes which are composed of many other simple attributes.

Example-

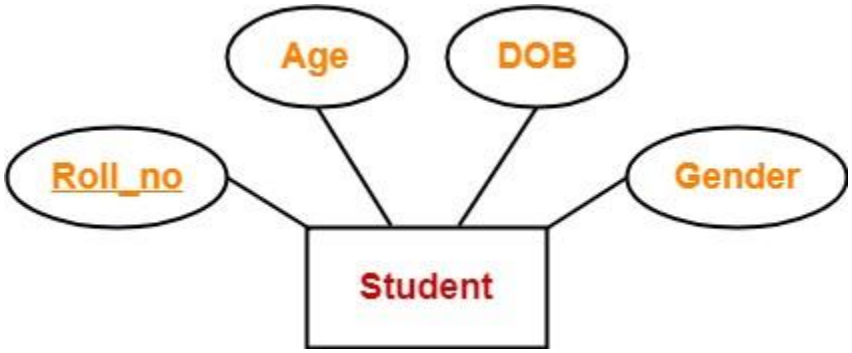


Here, the attributes "Name" and "Address" are composite attributes as they are composed of many other simple attributes.

Single Valued Attributes-

Single valued attributes are those attributes which can take only one value for a given entity from an entity set.

Example-

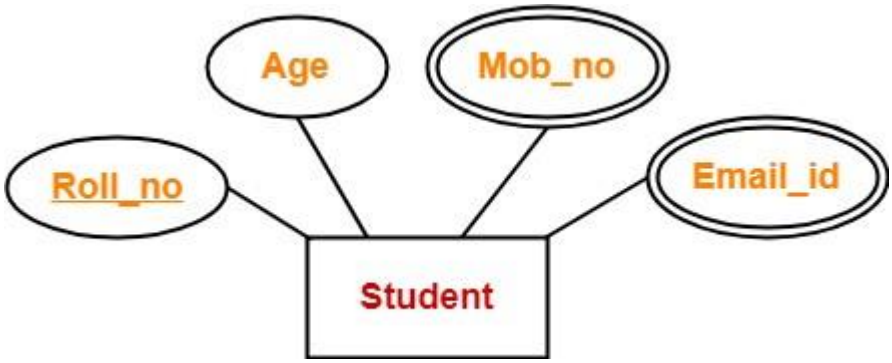


Here, all the attributes are single valued attributes as they can take only one specific value for each entity.

Multi Valued Attributes-

Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set.

Example-

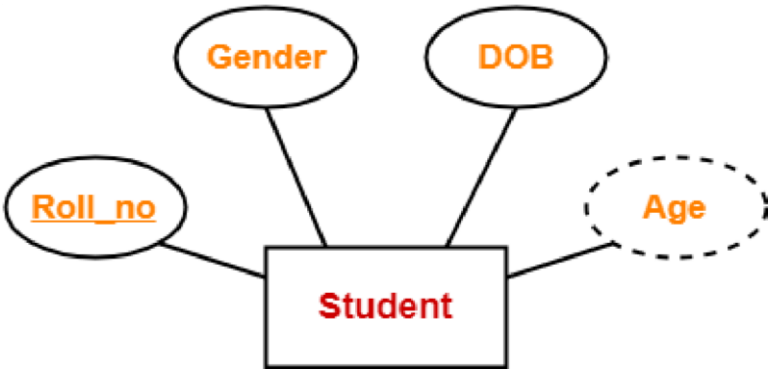


Here, the attributes "Mob_no" and "Email_id" are multi valued attributes as they can take more than one values for a given entity.

Derived Attributes-

Derived attributes are those attributes which can be derived from other attribute(s).

Example-



Here, the attribute "Age" is a derived attribute as it can be derived from the attribute "DOB".

Entity Set

An entity set is a **group of entities** that posses the **same set of attributes**. Each entity in an entity set has its **own set of values** for the **attributes** which make it distinct from other entities in a table. No two entities in an entity set will have the same values for the attributes.

In a database, an entity set is represented by the Table. Below you can see the Student Table which as multiple entries i.e. entity. Now, observe that the two students have the name Jhoson but, still they are uniquely identified as both posses different roll number.

Roll No.	Name	Course
CS08	Steive	Comp. Sci.
EE54	Jhoson	Electronics
B12	Eva	Biology
F32	Jhoson	Finance
M26	Erica	Maths

Student Table (Entity Set)

Well, in ER diagram an entity set is always represented with the rectangle. But, an **entity** can never be represented in ER diagram as it is just an **instance**.

KEYS

Keys are very important part of Relational database model. They are used to establish and identify relationships between tables and also to uniquely identify any record or row of data inside a table. A Key can be a single attribute or a group of attributes, where the combination may act as a key.

TYPES OF KEYS

SUPER KEY

A superkey is a set of attributes within a table whose values can be used to uniquely identify a tuple. A candidate key is a minimal set of attributes necessary to identify a tuple; this is also called a minimal superkey. ... employeeID is a candidate key.

FOREIGN KEY

A FOREIGN KEY is a key used to link two tables together. A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

Examples:
Table CUSTOMER

Column Name	Characteristic
SID	Primary Key
Last_Name	
First_Name	

Table ORDERS

Column Name	Characteristic
Order_ID	Primary Key
Order_Date	
Customer_SID	Foreign Key

Amount	

In the above example, the Customer_SID column in the ORDERS table is a foreign key pointing to the SID column in the CUSTOMER table.

PRIMARY KEY

A primary key is either an existing table column or a column that is specifically generated by the database according to a defined sequence.

For example, students are routinely assigned unique identification (ID) numbers, and all adults receive government-assigned and uniquely-identifiable Social Security numbers.

CANDIDATE KEY

The minimal set of attribute which can uniquely identify a tuple is known as candidate key.

For Example, STUD_NO in STUDENT relation. The value of Candidate Key is unique and non-null for every tuple. ... For Example, {STUD_NO, COURSE_NO} is a composite candidate key for relation STUDENT_COURSE.

ALTERNATE KEY

ALTERNATE KEYS is a column or group of columns in a table that uniquely identify every row in that table. A table can have multiple choices for a primary key but only one can be set as the primary key. All the keys which are not primary key are called an Alternate Key.

COMPOSITE KEY

composed of two or more attributes, but it must be minimal

RELATIONSHIP SET

A relationship set is a set of relationships of the same type. Formally it is a mathematical relation on (possibly non-distinct) sets.

TYPES OF RELATIONSHIP

Three types of relationship in database:

*Unary Relationship

*Binary Relationship

*Ternary Relationship

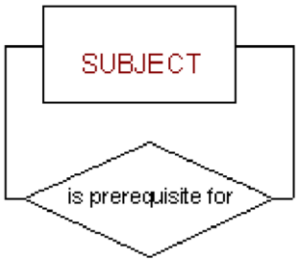
UNARY RELATIONSHIP

A unary relationship, also called recursive, is one in which a relationship exists between occurrences of the same entity set. In this relationship, the primary and foreign keys are the same, but they represent two entities with different roles.

For some entities in a unary relationship, a separate column can be created that refers to the primary key of the same entity set.

For Example:

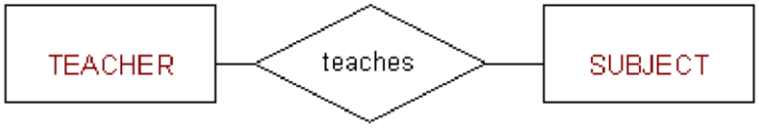
Subjects may be prerequisites for other subjects.



BINARY RELATIONSHIP

A Binary Relationship is the relationship between two different Entities , it is a relationship of role group of one entity with the role group of another entity.

A binary relationship is when two entities participate, and is the most common relationship degree.
For Example:

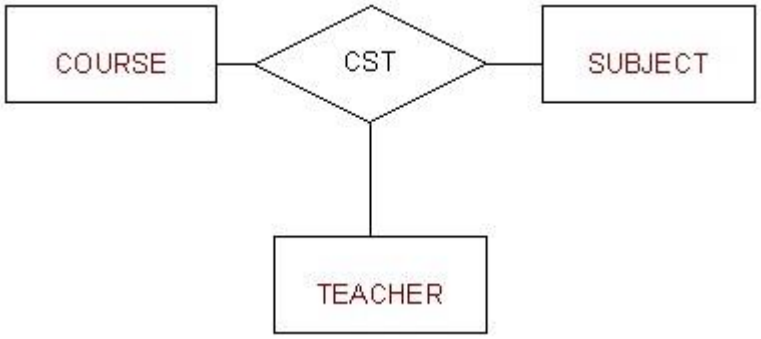


TERNARY RELATIONSHIP

A ternary relationship is when three entities participate in the relationship.

For Example:

The University might need to record which teachers taught which subjects in which courses.



TYPES OF BINARY RELATIONSHIP

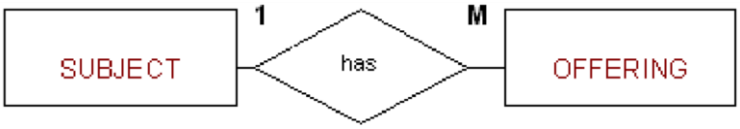
ONE TO ONE:

A Principal Teacher manages one Department

Each Department is managed by one Principal Teacher

ONE TO MANY:

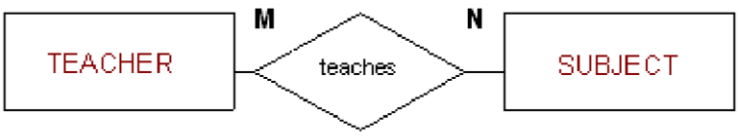
A Subject can be offered many times
Each Offering belongs to one Subject



MANY TO MANY

A Teacher can teach many different Subjects

Each Subject can be taught by many Teachers



Normalization

Normalization is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies.

Anomaly:

An error or inconsistency that may result when a user attempts to update a table that contains redundant data**.**

There are three types of Anomaly:

1. Insertion Anomaly,

2. Deletion Anomaly,

3. Modification Anomaly.

Normal Form Rules:

- **First Normal Form (1NF)**
- **Second Normal Form (2NF)**
- **Third Normal Form (3NF)**
- **Boyce-Codd Normal Form (BCNF)**
- **Fourth Normal Form (4NF)**

First Normal Form (1NF)

A relation that contains no multivalued Attributes.

Second Normal Form (2NF)

A relation in First Normal Form in which every attribute in fully functionally dependent in the primary key or Partial Functional dependency should be removed.

Partial Functional Dependency:

A functional dependency in which one or more non-key attribute are functionally dependent in part (but not all) of the primary key.

Functional Dependency:

A constrain between two attribute or two sets of attributes.

Third Normal Form (3NF)

A relation in Second Normal Form has no Transitive Dependency present.

Transitive Dependency

A Functional Dependency between two (or more) non-key attributes.

Boyce-Codd Normal Form(BCNF)

A relation in which every Determinant is a Candidate Key.

Determinant:

The Attributes or combination of Attributes that uniquely identifies a row in a relation.

<OR>

The Attribute on the left hand side of the arrow in a Functional Dependency.

Fourth Normal Form (4NF)

A relation in BCNF that contains no Multivalued Dependency.

Multivalued Dependency:

The type of dependency that exists when there are at least three Attributes in a relation with well-defined set of B and C values for each A value but those B and C values are independent each other.

JOIN

A join is an SQL operation performed to establish a connection between two or more database tables based on matching columns, thereby creating a relationship between the tables.

Different types of Joins are:

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

Relational Algebra

Relational Algebra is procedural query language, which takes Relation as input and generate relation as output. Relational algebra mainly provides theoretical foundation for relational databases and SQL.

Operators in Relational Algebra:

Projection (π)
Projection is used to project required column data from a relation.

Selection (σ)

Selection is used to select required tuples of the relations.

Union (U)
Union operation in relational algebra is same as union operation in set theory, only constraint is for union of two relation both relation must have same set of Attributes.

Set Difference (-)
Set Difference in relational algebra is same set difference operation as in set theory with the constraint that both relation should have same set of attributes.

Rename (ρ)
Rename is a unary operation used for renaming attributes of a relation.
ρ (a/b)R will rename the attribute ‘b’ of relation by ‘a’.

Cross Product (X)
Cross product between two relations let say A and B, so cross product between A X B will results all the attributes of A followed by each attribute of B. Each record of A will pairs with every record of B.

Natural Join (⋈)

Natural join is a binary operator. Natural join between two or more relations will result set of all combination of tuples where they have equal common attribute.

Conditional Join
Conditional join works similar to natural join. In natural join, by default condition is equal between common attribute while in conditional join we can specify the any condition such as greater than, less than, not equal.

Data Control Language (DCL)

DCL commands are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE.

SQL GRANT Command:

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

SQL REVOKE Command:

The REVOKE command removes user access rights or privileges to the database objects.

Privileges and Roles:

Privileges: Privileges defines the access rights provided to a user on a database object. There are two types of privileges.

1. System privileges - This allows the user to CREATE, ALTER, or DROP database objects.

2. Object privileges - This allows the user to EXECUTE, SELECT, INSERT, UPDATE, or DELETE data from database objects to which the privileges apply.

Roles: Roles are a collection of privileges or access rights. When there are many users in a database it becomes difficult to grant or revoke privileges to users.

VIEW: 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.

SQL QUERY

01. CREATE DATABASE:

```
CREATE DATABASE
databasename;
```

02. DROP DATABASE:

```
DROP DATABASE databasename;
```

03. CREATE TABLE:

```
CREATE TABLE table_name (
column1 datatype(size),    column2
datatype(size),    column3
datatype(size)

);
```

04. INSERT VALUES:

- INSERT INTO table_name (column1, column2, column3, ...)
  VALUES (value1, value2, value3, ...);
- INSERT INTO table_name
  VALUES (value1, value2, value3, ...);

05. UPDATE:

```
UPDATE table_name
SET column1 = value1, column2 = value2,
```

WHERE condition;

06. DELETE:

DELETE FROM *table_name* WHERE *condition*;

07. SELECT:

- SELECT column1, column2, ...
  FROM table_name;

- SELECT * FROM table_name;

08. DROP:

DROP TABLE *table_name*;

09. ORDER BY:

SELECT *column1*, *column2*
FROM *table_name*
ORDER BY *column1*, *column2,* ASC|DESC;

10. WHERE:

SELECT column1, column2,

FROM table_name WHERE

condition;

11. Alias:

SELECT column_name AS alias_name

FROM table_name;

12. IN:

- SELECT column_name(s)
  FROM table_name
  WHERE column_name IN (value1, value2,);

- SELECT column_name(s)
  FROM table_name
  WHERE column_name IN (SELECT STATEMENT);

13. COUNT:

SELECT COUNT(column_name)

FROM table_name WHERE

condition;

14. SUM:

SELECT SUM (column_name)

FROM table_name

WHERE condition;

15. GROUP BY:

SELECT *column_name*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name;*

16. MAXIMUM:

SELECT MAX(column_name)
FROM table_name WHERE
condition;

17. MINIMUM:

SELECT MIN(*column_name*)
FROM *table_name*
WHERE *condition*;

18. AVERAGE:

SELECT AVG(*column_name*)
FROM *table_name*
WHERE *condition*;

19. BETWEEN:
SELECT *column_name*
FROM *table_name*
WHERE *column_name* BETWEEN *value1* AND *value2;*

20. LIKE:

SELECT *column1*, *column2,*
FROM *table_name*
WHERE *columnN* LIKE *pattern*;

21. AND:

SELECT *column1*, *column2,*
FROM *table_name*
WHERE *condition1* AND *condition2* AND *condition3* ;

22. OR:

SELECT *column1*, *column2,*
FROM *table_name*
WHERE *condition1* OR *condition2* OR *condition3* ;

23. NOT:

SELECT *column1*, *column2,*
FROM *table_name*
WHERE NOT *condition*;

24. ROWNUM:

SELECT *column_name(s)*
FROM *table_name*
WHERE ROWNUM <= *number*;

25. SELECT TOP:

- SELECT TOP number|percent column_name(s)
  FROM table_name
  WHERE condition;

- SELECT column_name(s)
  FROM table_name
  WHERE ROWNUM <= number;

- SELECT column_name(s)
  FROM table_name
  WHERE condition
  LIMIT number;

26. NULL:

- SELECT column_names FROM
  table_name
  WHERE column_name IS NULL;

- SELECT column_names

FROM table_name WHERE
column_name IS NOT NULL;

27. Constraints:

CREATE TABLE table_name (

column1 datatype constraint, column2

datatype constraint, column3 datatype

constraint,

);

28. UNION:

- SELECT column_name(s) FROM table1
  UNION
  SELECT column_name(s) FROM table2;

- SELECT column_name(s) FROM table1
  UNION ALL
  SELECT column_name(s) FROM table2;

29. INNER JOIN:

SELECT column_name(s)

FROM table1

INNER JOIN table2

ON table1.column_name = table2.column_name;

30. LEFT JOIN:

SELECT *column_name(s)*
FROM *table1*
LEFT JOIN *table2*
ON *table1.column_name = table2.column_name*;

31. RIGHT JOIN:

SELECT column_name(s)

FROM table1
RIGHT JOIN table2

ON table1.column_name = table2.column_name;

32. SELF JOIN:

SELECT column_name(s)

FROM table1 T1, table1 T2

WHERE condition;

33. OUTER JOIN:

SELECT *column_name(s)*
FROM *table1*
FULL OUTER JOIN *table2*
ON *table1.column_name = table2.column_name*
WHERE *condition*;

34. HAVING:

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
GROUP BY *column_name(s)*
HAVING *condition* ORDER
BY *column_name(s);*

35. ANY:

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name operator* ANY
(SELECT *column_name* FROM *table_name* WHERE *condition*);

36. ALL:

SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name operator* ALL
(SELECT *column_name* FROM *table_name* WHERE *condition*);

37. CASE:

CASE
  WHEN *condition1* THEN *result1*
  WHEN *condition2* THEN *result2*
  WHEN *conditionN* THEN *resultN*
  ELSE *result*
END;

38. EXISTS:

SELECT column_name(s)

FROM table_name

WHERE EXISTS

(SELECT column_name FROM table_name WHERE condition);

39. SELECT INTO:

- SELECT *
  INTO newtable [IN externaldb]
  FROM oldtable
  WHERE condition;

- SELECT column1, column2, column3, ...
  INTO newtable [IN externaldb]
  FROM oldtable WHERE
  condition;

40. INSERT INTO:

- INSERT INTO table2
  SELECT * FROM table1
  WHERE condition;

- INSERT INTO table2 (column1, column2, column3, ...) SELECT
  column1, column2, column3, ...
  FROM table1 WHERE

condition;

41. BACKUP:

BACKUP DATABASE *databasename*
TO DISK = '*filepath*';

42. ADD:

ALTER TABLE *table_name*
ADD *column_name datatype*;

43. DROP:

ALTER TABLE *table_name*
DROP COLUMN *column_name*;

44. ALERT COLUME:

ALTER TABLE *table_name*
ALTER COLUMN *column_name datatype*;

45. MODIFY:

ALTER TABLE *table_name*
MODIFY COLUMN *column_name datatype*;

46. CREATE INDEX:

CREATE INDEX *index_name*
ON *table_name* (*column1*, *column2*);

47. CREATE UNIQUE INDEX:

CREATE UNIQUE INDEX *index_name*
ON *table_name* (*column1*, *column2*);

48. CREATE SEQUENCE:

CREATE SEQUENCE [schema.]sequence_name
[AS datatype]
[START WITH value]
[INCREMENT BY value]
[MINVALUE value | NO MINVALUE]
[MAXVALUE value | NO MAXVALUE]
[CYCLE | NO CYCLE]
[CACHE value | NO CACHE];

49. VARIANCE:

select VARIANCE(char_length) from
all_tab_columns;

50. CREATE VIEW: CREATE

VIEW view_name AS SELECT

column1, column2, ...

FROM table_name

WHERE condition; **UPDATE**

VIEW:

Update view_name
Set column_Name

Where condition; **INSERT**

INTO:

INSERT INTO view_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

DELETE:

DELETE FROM view_name

WHERE conditions; **DROP**

VIEW:

DROP VIEW view_name;

51. USERS:

CREATE USERS user_name indentified by password;

ALTER USERS user_name indentified by password;

52. GRANT:

- GRANT privilege_name
  ON object_name
  TO {user_name |PUBLIC |role_name}
  [WITH GRANT OPTION];
- GRANT CREATE TABLE TO testing;
- GRANT testing TO user1;

53. REVOKE:

- REVOKE privilege_name
  ON object_name
  FROM {user_name |PUBLIC |role_name} •
  REVOKE CREATE TABLE FROM testing;

54. ROLE:

- CREATE ROLE role_name
  [IDENTIFIED BY password];
- CREATE ROLE testing
  [IDENTIFIED BY pwd];
- DROP ROLE role_name;
- DROP ROLE testing;

****