11/4/23

CHAPTER
RELATIONAL DATABASE AND SQL

☆ DATABASE

A database is defined as an organized collection of data (information) about an entity (something that exists) or things.

DATABASE
- Integrated
- Shared

- **DATA**

Data / character is the smallest unit of file organization which represents itself in the form of a bit that may be either 0 or 1.

- **FIELD / DATA ITEM**

A field is a set of characters which are used together to represent specific data elements. For example, Roll number, Name, Age and Marks are the fields in a Student's record.

- **RECORD**

A collection of fields is termed as a record. For example, a Student record consists of the field Roll No, Name, Age and Marks.

- **FILE / TABLE / RELATION**

A collection of logically related records is called a file.

TABLE
- Rows (Records/tuples)
- Columns (Attributes/fields)

- **DEGREE**

Number of fields in a table is the degree of that table.

- CARDINALITY

Number of records in a table is the cardinality of that table.

Fields / data-items / attributes

ROLL NO.	NAME	AGE	MARKS
20	Sonia	14	90

→ Record / tuple

Data

Degree = 4
Cardinality = 1

★ ADVANTAGES OF A DBMS → A DBMS is a general purpose software system that facilitates the process of defining, constructing and manipulating databases for various applications.

- ELIMINATION OF DATA REDUNDANCY

Duplication of data leads to wastage in storage space. A DBMS eliminates data redundancy (duplication of data) by integrating the files so that multiple copies of the same data are not stored.

- DATA CONSISTENCY

A DBMS provides data consistency to a large extent as the changes made at one place are reflected at all other places or to all the users.

- SHARING OF DATA

By using a DBMS, not only can existing applications share data in the database, but new applications can also be developed to operate against the same stored data.

- REDUCED PROGRAMMING EFFORT

A DBMS saves a lot of programming effort since a user need not write programs for query processing involving several tables or files, report generation, addition, modification and deletion of data, etc. Thus, it provides easy retrieval

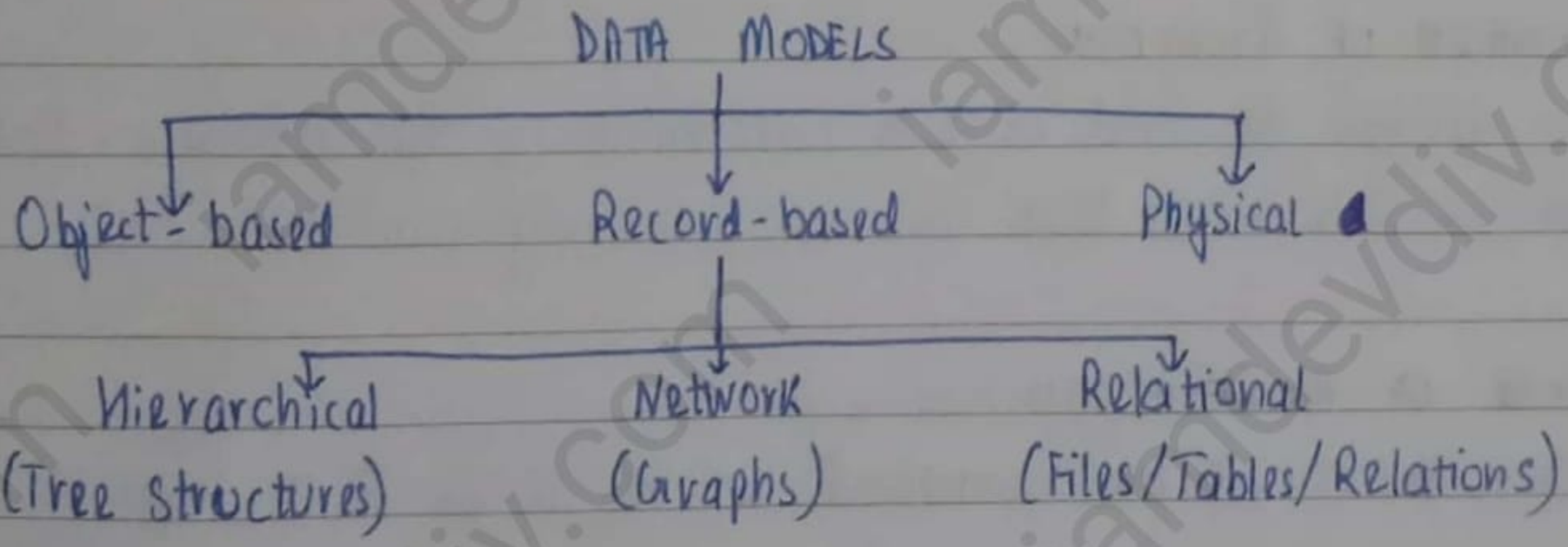
of data.

- IMPROVED DATA INTEGRITY
  Data integrity refers to the validity and consistency of stored data. For example, the system itself checks for the correct information to be entered by the user in the correct format.

- PRIVACY AND SECURITY
  Data security refers to the protection of data against accidental or intentional disclosure to unauthorized persons.

- IMPROVED BACKUP AND RECOVERY SYSTEM
  A database system provides facilities for recovery from hardware or software failures.

☆ DBMS MODELS



- RELATIONAL DATABASE
  A relational database is a type of database that stores and provides access to data points that are related to one another.

→ BASIC TERMINOLOGIES

1. ENTITY
   An entity is something that exists and about which we can store some information. For example, student entity, employee entity, etc.

2. ATTRIBUTE

An attribute is a set of values of a particular type. A table consists of several records (rows); each record can be broken into several smaller entities known as fields or attributes or columns.

3. TUPLE

Each row in a table is known as tuple. It is also called a row/record. A single entry in a table is called a record or row. A record in a table represents a set of related data.

4. CARDINALITY OF RELATION

It is the number of records or tuples in the relation.

5. DEGREE OF RELATION

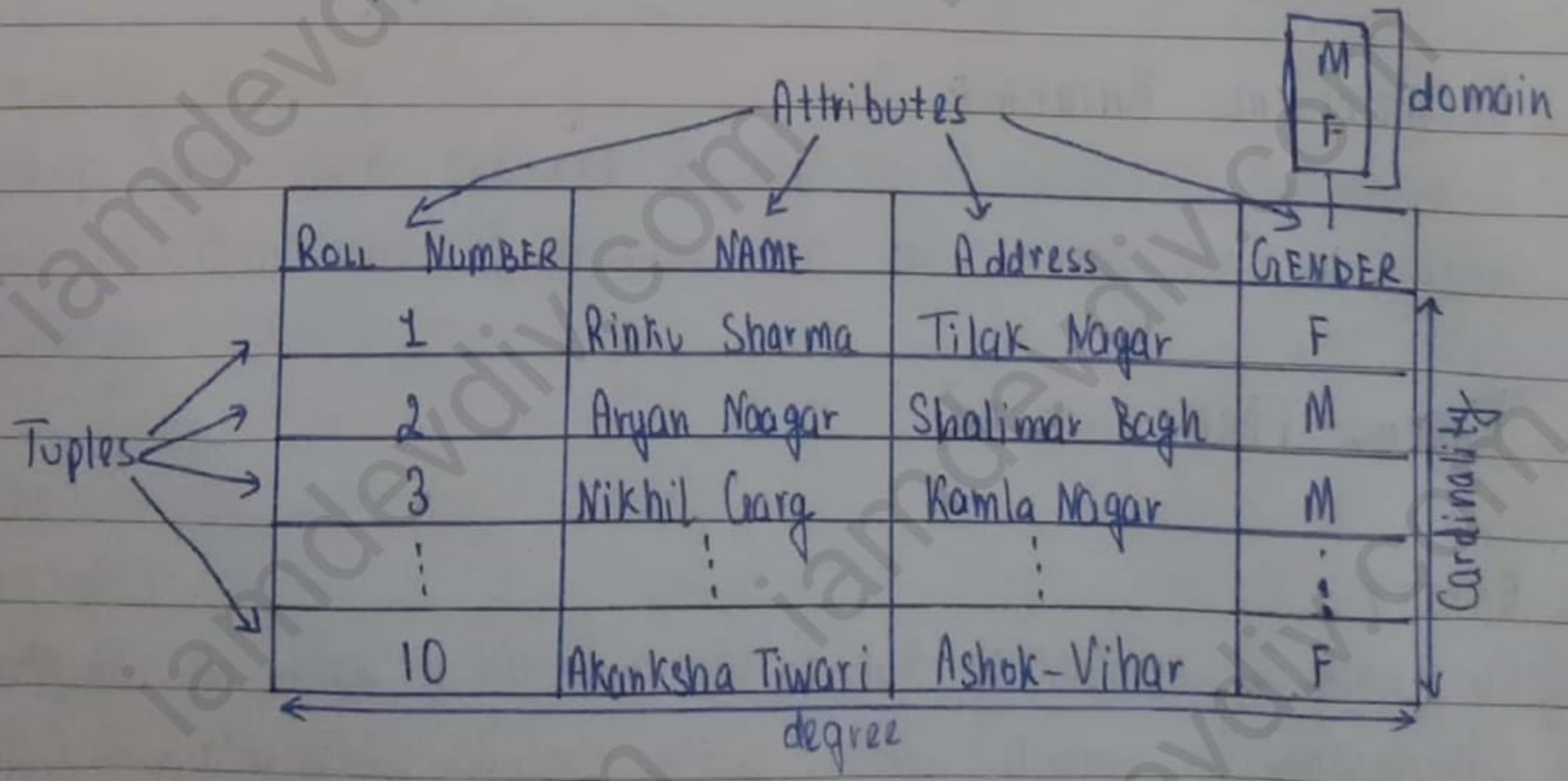
Number of columns or attributes is known as degree of relation.

6. DOMAIN OF RELATION

It defines the kind of data represented by the attribute. It is the set of all possible values that an attribute may contain.

7. BODY OF A RELATION

It consists of an unordered set of 0 or more tuples.



★ DATABASE KEYS

Keys allow us to identify an attribute or a set of attributes on the basis of which a table is identified. They are used to establish and identify relation among two or more tables.

1. PRIMARY KEY

A primary key is a set of one or more attributes/fields which uniquely identifies a tuple/row in a table.

(a) It must contain unique values, i.e., ~~jo~~ non-redundant.
(b) It arranges the table in its own order.
(c) It cannot be re-declared or left null.
(d) One table can have only one primary key; however, primary key can be a combination of more than one field.

2. CANDIDATE KEY

A candidate key refers to all the attributes in a relation that are candidates or are capable of becoming a primary key.

Candidate keys − Primary key = Alternate key

3. ALTERNATE KEY

A candidate key that is not the primary key is called an alternate key. In other words, any attribute that is a candidate for the primary key, i.e., which is capable of becoming a primary key but is not a primary key, is an alternate key.

4. FOREIGN KEY

A foreign key is a non-key attribute whose value is derived from the primary key of another table; in other words, a primary key in some other table having relationship with the current or original table.

This is to keep in mind that as per the Referential integrity constraint, any value which does not exist as a primary key in the parent table cannot

be taken in the child table.

- REFERENTIAL INTEGRITY

Referential integrity is a constraint in the database that enforces the relationship between two tables. The Referential Integrity constraint requires that values in a foreign key column must either be present in the primary key that is referenced by the foreign key or they must be null.

★ INTRODUCTION TO SQL

* All the real-life applications require a DBMS to manipulate and handle ~~this~~ enormous data. A DBMS requires some language to handle and manipulate its data, which is known as Structured Query Language (SQL).

* SQL is a standard language for accessing and manipulating databases. SQL commands are used to create, transform and retriere information from Relational Database Management Systems and are also used to create interface between a user and a database.

- MySQL

* MySQL is an open-source and freely-available Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).

* It provides excellent features for creating, storing, maintaing and accessing data, stored in the form of databases and their respective tables.

- FEATURES OF SQL

- retriere data through query processing
- insert records
- update records
- create new databases and modify the existing ones
- create new tables
- create views
- modifying the security settings of the system

- ADVANTAGES OF SQL

1. EASE OF USE

It is very easy to learn and use and does not require high-end professional training to work upon it.

2. Large volume of databases can be handled quite easily.

3. NO CODING REQUIRED

It is non-procedural and a unified language, i.e., we need not specify the procedures to accomplish a task but only need to give a command to perform the activity.

4. SQL can be linked to most of the other high-level languages.

5. PORTABLE

It is compatible with other database programs like DBase IV, FoxPro, etc.

6. SQL is ~~not~~ a case-insensitive language.

☆ CLASSIFICATION OF SQL STATEMENTS

- DATA DEFINITION LANGUAGE (DDL) COMMANDS

The DDL part of SQL permits database tables to be created or

deleted. It contains necessary statements for creating, manipulating, altering and deleting the table.
Examples of DDL commands in SQL are:

* CREATE DATABASE: creates a new database.
* USE: to select and open an already existing database.
* SHOW: to display all the tables in an existing database.
* CREATE TABLE: creates a new table in an existing database.
* ALTER TABLE: modifies the structure of a table.
* DROP TABLE: deletes a table.

- DATA MANIPULATION LANGUAGE (DML) COMMANDS

DML is a part of SQL that helps a user manipulate data. DML commands carry out query processing operations and manipulate data in the database objects.

1. INSERT INTO: To insert new data (record) into a table.
2. UPDATE: To modify or change the data (tuple) in a table.
3. DELETE: To delete data (tuple) from a table.

→ DIFFERENCE BETWEEN DDL and DML COMMANDS

DDL Commands	DML Commands
1. DDL stands for Data Definition language	1. DML stands for Data Manipulation language.
2. These commands allow us to perform tasks related to data definition, i.e, related to the structure of the database objects.	2. These commands are used to manipulate data, i.e, records or rows in a table or relation.
3. The examples of DDL commands are Create, Alter, Drop, etc.	3. The examples of DML commands are Insert into, Update, Delete, etc.

- DATA QUERY LANGUAGE (DQL) - SELECT COMMAND

The SELECT command is used to query or retrieve data from a table in the database. It is used to retrieve a subset of records from one or more tables.

★ ADVANTAGES OF My SQL

1. Reliability and Performance
2. Modifiable
3. Multi - Platform Support
4. Powerful Processing Capabilities
5. Integrity (Checks)
6. Authorization (restricting access)
7. Powerful Language (operations are performed at a prescribed and fixed level)
8. Reliable (well-defined set of commands)
9. Freedom of Data abstraction
10. Complete Language for a Database

★ SQL DATA TYPES

- INTEGER

Positive whole numbers upto 11 digits and negative whole numbers up to 10 digits

- FLOAT

Holds numbers with decimal points. Occupies 4 bytes.

- NUMERIC[redacted] (x,y) → holds upto 20 significant digits

x = total number of digits

y = the number of places to the right of the decimal point

For example, Numeric (8,2) ⇒ 5 places before the decimal
2 digits after the decimal
1 digit place for decimal point

A negative number holds one place for the sign, i.e., (-).

- DECIMAL (x, y) → holds up to 19 significant digits
  x = total number of digits
  y = the number of places to the right of the decimal point.
  For example, Decimal(8,2) ⇒ 6 digits before the decimal
  2 digits after the decimal
  A negative number uses one place for its sign (-).

- CHAR ~~(size)~~ (x) → maximum of 254 characters (single or double quotation marks)
  Stores 'x' number of characters in the string which has a fixed length.

- VARCHAR(x) → single or double quotation marks
  Stores variable length alphanumeric data.

- DATE → single quotation marks
  Stores a date in 'yyyy/mm/dd' or 'yyyy-mm-dd' format.

- TIME
  Stores time in hh:mm:ss format.

→ DIFFERENCE BETWEEN CHAR AND VARCHAR DATA TYPES

CHAR	VAR
1. Fixed length memory storage, fixed length character string.	1. Variable length memory storage, variable length character string.
2. Can store a maximum of 0 to 255 characters	2. Can store a maximum number up to 65,535.
3. CHAR(x) will take x characters of storage even if you enter less than x characters.	3. VARCHAR(x) will take only required storage for the actual number of characters entered.
4. If a value entered is shorter ~~even~~ than	4. No blanks are added if the length

its length x, then blanks are ~~if you enter enter less than x~~ added.	4. is shorter than the maximum length, x.
5. Takes memory space of 1 byte per character.	5. Takes memory space of 1 byte per info. character, +2 bytes to hold variable length
6. Search operation is faster.	6. Search operation is slower.
7. For example, name char(10); name := "anu"; name field occupies 10 bytes, with the first three bytes with values and the rest with blank data.	7. For example, name varchar(10); name = "anu"; then name occupies only 3+2 = 5 bytes, the first three bytes for value and the other two bytes for variable length info.

☆ SQL COMMANDS

* EXAMPLE 1

- sql > SHOW DATABASES; ↴ → Press Enter key
- sql > CREATE DATABASE abc; ↴
- sql > USE abc; ↴
- sql > SHOW TABLES; ↴
- sql > SELECT DATABASE(); ↴

- sql > CREATE TABLE emp ↴
  > (e_no CHAR(5) PRIMARY KEY, ↴
  > e_name VARCHAR(25) NOT NULL, ↴
  > age INTEGER, ↴
  > city VARCHAR(25) DEFAULT "SRE", ↴
  > salary DECIMAL); ↴

- sql > INSERT INTO emp ↴
  > VALUES ("E0001", "Ravi", 32, "Delhi", 50000); ↴
- sql > INSERT INTO emp(e_no, e_name, salary) ↴
  > VALUES("E0002", "Ajay", 25000); ↴

- sql > ALTER TABLE emp ↴
  > ADD (mobile_no CHAR(10)); ↴
  ↳ can add multiple fields separated with commas at a time

- sql > ALTER TABLE emp ↵
  > MODIFY (e_name VARCHAR(40)); ↵
  ↳ can only modify single field at a time
- sql > ALTER TABLE emp ↵
  > DROP (mobile_no); ↵
  ↳ can only delete single field at a time
- sql > DESC emp; ↵ (OR) DESCRIBE emp; ↵
- sql > SELECT * FROM emp; ↵
- sql > ALTER TABLE emp CHANGE city state VARCHAR(20); ↵

- sql > UPDATE emp ↵
  > SET salary = salary + salary * 0.05 ↵
  > WHERE age > 55; ↵
- sql > DELETE FROM emp ↵
  > WHERE age > 60; ↵

* EXAMPLE 2

- sql > CREATE DATABASE div_db;
- sql > USE div_db;
- sql > CREATE TABLE div_table(
  > e_no CHAR(5) PRIMARY KEY,
  > e_name VARCHAR(20) NOT NULL,
  > desig VARCHAR(15),
  > salary FLOAT);
- sql> INSERT INTO div_table VALUES(
  > 'E0001', 'Raghav Pundir', 'MANAGER', 50000);
- sql > INSERT INTO div_table VALUES(
  > 'E0002', 'Aryan Kumar', 'ADMINISTRATOR', 45000);
- sql > INSERT INTO div_table VALUES(
  > 'E0003', 'Arjun Panchal', 'CEO', 100000);
- sql > INSERT INTO div_table VALUES(
  > 'E0004', 'Sanjay Sharma', 'ASSISTANT', 42000);

- sql > INSERT INTO div_table VALUES(
  > 'E0005', 'Aayushi Verma', 'RECEPTIONIST', 30000);
- sql > ALTER TABLE div_table ADD(city VARCHAR(20));
- sql > UPDATE ~~TABLE~~ div_table SET city = 'SRE' WHERE e_no = 'E0001';
- sql > UPDATE div_table SET city = 'Delhi' WHERE e_no = 'E0002';
- sql > UPDATE div_table SET city = 'Roorkee' WHERE e_no = 'E0003';
- sql > UPDATE div_table SET city = 'SRE' WHERE e_no = 'E0004';
- sql > UPDATE div_table SET city = 'Jaipur' WHERE e_no = 'E0005';
- sql > DELETE FROM div_table WHERE city = 'SRE';
- sql > DESC div_table;
- sql > SELECT DATABASE();
- sql > ALTER TABLE div_table DROP salary;

- SQL QUERY PROCESSING

mysql > SELECT * FROM emp; ↵
mysql > SELECT e_name, salary FROM emp; ↵
mysql > SELECT * FROM emp WHERE salary > 50000; ↵
mysql > SELECT city FROM emp; ↵
mysql > SELECT DISTINCT city FROM emp; ↵
mysql > SELECT salary * 12 'Annual salary' FROM emp; ↵
mysql > SELECT * FROM emp WHERE salary <> 50000; ↵
↳ not equals to
mysql > SELECT * FROM emp WHERE ↵
> desig = 'MANAGER' OR salary < 50000; ↵
mysql > SELECT * FROM emp WHERE NOT (desig = 'ADMINISTRATOR'); ↵
mysql > SELECT * FROM emp WHERE age BETWEEN 40 AND 50; ↵
mysql > SELECT * FROM emp WHERE age ≤ 40 AND age ≥ 50; ↵
mysql > SELECT * FROM emp WHERE desig IN ('MANAGER', 'CLERK', 'SUPERVISOR

* WILD CARD CHARACTERS
(i) % ⇒ Zero or more than zero characters
(ii) _ ⇒ Exactly one character

```
mysql > SELECT * FROM emp WHERE e-name LIKE 'A%'; ↵
mysql > SELECT * FROM emp WHERE e-name LIKE '_a%'; ↵
mysql > SELECT * FROM emp WHERE e-name LIKE 'Ra__'; ↵
mysql > SELECT * FROM emp WHERE e-name LIKE '%h'; ↵
mysql > SELECT * FROM emp WHERE e-name LIKE '%h%'; ↵
mysql > SELECT * FROM emp WHERE e-name LIKE '%_r'; ↵
```

* ALIASING IN SQL

```
mysql > SELECT e-name AS "Employee Name" FROM emp; ↵
```

* SORTING IN SQL

```
mysql > SELECT * FROM emp ORDER BY e-name ASC; ↵
                                  ↳ default
mysql > SELECT * FROM emp ORDER BY e-name ASC, salary DESC; ↵
```

Q To show all the records of emp table for those employees who are manager in ascending order.

Ans. `SELECT * FROM emp WHERE desig = 'MANAGER' ORDER BY e-name; ↵`

• LAB PRACTICE

```
mysql > CREATE DATABASE divyanshu_db;
mysql > USE divyanshu_db;
mysql > CREATE TABLE emp (
    > e_no CHAR(5) PRIMARY KEY,
    > e-name VARCHAR(20) NOT NULL,
    > desig VARCHAR(15),
    > age INTEGER,
    > salary INTEGER);
mysql > DESC emp;
```

Output on next page ⟶

Field	Type	NULL	Key	Default	Extra
e-no	char(5)	NO	PRI	NULL	
e-name	varchar(20)	NO		NULL	
desig	varchar(15)	YES		NULL	
age	int	YES		NULL	
salary	int	YES		NULL	

```
mysql > INSERT INTO emp VALUES(
    > "E0001", "Divyanshu Tiwari", "Programmer", 17, 80000);
mysql> INSERT INTO emp VALUES(
    > "E0002", "Aayush Saini", "Designer", 17, 75000);
mysql> INSERT INTO emp VALUES(
    > "E0003", "Aviral Sharma", "Story Writer", 17, 85000);
mysql> INSERT INTO emp VALUES(
    > "E0004", "Himanshu Chauhan", "CMO", 18, 100000);
mysql > INSERT INTO emp VALUES(
    > "E0005", "Shirupiru Gagneja", "Tester", 17, 70000);
mysql> INSERT INTO emp VALUES(
    > "E0006", "Aviral Gupta", "Music Producer", 17, 90000);
mysql> INSERT INTO emp VALUES(
    > "E0007", "Om Panchal", "Tester", 17, 70000);
```

```
mysql > SELECT * FROM emp;
```

e-no	e-name	desig	age	Salary
E0001	Divyanshu Tiwari	Programmer	17	80000
E0002	Aayush Saini	Designer	17	75000
E0003	Aviral Sharma	Story Writer	17	85000
E0004	Himanshu Chauhan	CMO	18	100000
E0005	Shirupiru Gagneja	Tester	17	70000
E0006	Aviral Gupta	Music Producer	17	90000
E0007	Om Panchal	Tester	17	70000

mysql > ALTER TABLE emp ADD city VARCHAR(15) DEFAULT "Saharanpur";
mysql > UPDATE emp SET city = "Delhi" WHERE desig = "Tester";
mysql > SELECT e-name, city FROM emp;

e-name	city
Divyanshu Tiwari	Saharanpur
Aayush Saini	Saharanpur
Aviral Sharma	Saharanpur
Himanshu Chauhan	Saharanpur
Shirupiru Gagneja	Delhi
Aviral Gupta	Saharanpur
Om Panchal	Delhi

mysql > UPDATE emp SET salary = salary + salary * 0.05 WHERE desig = "Tester";
mysql > SELECT e-name, salary FROM emp WHERE desig = "Tester";

e-name	Salary
Shirupiru Gagneja	73500
Om Panchal	73500

mysql > SELECT DISTINCT age FROM emp;

age
17
18

mysql > SELECT e-name, salary FROM emp WHERE salary > 80000 ORDER BY e-name;

e-name	salary
Aviral Gupta	90000
Aviral Sharma	85000
Himanshu Chauhan	100000

mysql > SELECT e-name FROM emp WHERE salary BETWEEN 70000 AND 75000;

e-name

Output on next page ⟶

e-name
Aayush Saini
Shirupiru Gagneja
Om Panchal

mysql> SELECT e-name AS "Employee Name" FROM emp WHERE e-name LIKE "A%";

Employee Name
Aayush Saini
Aviral Sharma
Aviral Gupta

mysql> SELECT e-name AS "Employee name ending with i" WHERE e-name LIKE "% i";

Employee names ending with i
Divyanshu Tiwari
Aayush Saini

mysql > SELECT e-name FROM emp WHERE age <> 17;

e-name
Himanshu Chauhan

* AGGREGATE FUNCTIONS

mysql> SELECT SUM (salary) FROM emp WHERE desig = "Manager";
mysql> SELECT COUNT (*) FROM emp WHERE salary < 50000;
mysql> SELECT COUNT (salary) FROM emp WHERE salary < 50000;
mysql> SELECT desig, SUM (salary), COUNT (*) FROM emp GROUP BY desig HAVING COUNT (*) > 2;

Relational Database and SQL continue after page no. 17

- SQL JOINS

An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

* CARTESIAN PRODUCT (CROSS PRODUCT)

a) The Cartesian product is a binary operation and is denoted by (x).
b) The degree of the new relation formed is the sum of the degrees of two relations on which Cartesian product is ~~formed~~ performed.

Rollno	Name
1	Rohan
2	Jaya
3	Teena

Table: Student

gameno	gname
10	Football
11	Lawn Tennis

Table: Games

Cartesion product for Student x Games:

```
mysql> SELECT Name, gname FROM Student, Games;
```

Student X Games

Name	gname
Rohan	Football
Jaya	Football
Teena	Football
Rohan	Lawn Tennis
Jaya	Lawn Tennis
Teena	Lawn Tennis

* EQUI JOIN

An Equi join is a simple SQL join condition that uses the equal to sign (=) as a comparison operator for defining a relationship between two tables on the basis of a common field, i.e. primary key and foreign key.

Rollno	Name
1	Rohan
2	Jaya
3	Teena
4	Diksha

Table: Student

Rollno	Fee
4	4500
2	5500
3	5000

Table: Fees

mysql> SELECT A.Rollno, A.Name, B.Fee FROM Student A, Fees B WHERE A.Rollno = B.Rollno;

Resultant Table

Rollno	Name	Fee
2	Jaya	5500
3	Teena	5000
4	Diksha	4500

* NATURAL JOIN

a) The JOIN in which only one of the identical columns exists is called Natural Join.

b) It is similar to Equi Join except that duplicate columns are eliminated in Natural Join that would otherwise appear in Equi Join.

c) In NATURAL JOIN, the join condition is not required; it automatically joins based on the common column value.

The example command given above for equi join can be written with natural join as:

mysql> SELECT * FROM Student NATURAL JOIN Fees;