Member1 Name: Himanshu Chauhan 2021323
Member2 Name: Ankit Kumar Singh 2021310

Showcase adequate relational algebraic operations

Relational Algebraic Operations for an **Online Pharmacy Store**:

SELECTION:

This operation is used to retrieve only the entries in a table that satisfy specific requirements. For instance, we could use SELECT to get a list of all medicine by a particular customer.

σ Cname = 'Ankit' (customer ⋈ medicine)

By executing a join between the Customer and Invoice tables and then choosing only the rows where the Customer name is "Ankit", this method would yield all invoice by the customer with the name Ankit.

PROJECTION:

A subset of columns from a table can be retrieved using this operation, or project. For instance, we might use PROJECT to retrieve the Doctor table's Dname and Speciality fields only.

Π Dname, Speciality (doctor)

Only the Dname and Speciality columns from the doctor table would be returned by this action.

JOIN:

This procedure combines rows from multiple tables that share a column. For instance, we could join the customer and invoice tables to obtain a list of every order and the relevant order details.

customer ⋈ invoice

On the basis of the shared column CID, this operation would conduct an inner join between the customer and invoice tables, returning a table with columns from both tables.

UNION:

This procedure is used to merge duplicate-free rows from two tables into one table. A table of customer and a table of doctor, for instance, may be combined using UNION:

customer ∪ doctor

A table containing all the rows from both tables, without any duplicates, would be produced by this process.

DIFFERENCE:

This operation is used to extract rows from one table from another table where they do not exist. Using DISTINCT, for instance, we could obtain a list of all the pharmacy that have state is equal to NULL.

pharmacy - (pharmacy ⋈ NULL)

By doing this action, all of the items in the pharmacy table would be subtracted from the pharmacy table, creating a list of pharName that have state is equal to NULL.

INTERSECTION:

This operation is used to find the common rows between two queries that have the same number and type of columns.

medicine ⋂ invoice

By doing this action, all the medID which is common from medicine and invoice table would be produced.

RENAMING:

This operation is used to rename a table or a column. For example, if you want to change the name of a column in the medicine table from "Mname" to "medicine_name", you could use the following SQL query:

SELECT Mname AS medicine_name FROM medicine;

Showcase our constraints

NOT NULL:

This restriction is used to state that NULL values are not permitted in a column. For example, the customer table's Cname column shouldn't be NULL.

```
CREATE TABLE `customer` (
  `CID` int NOT NULL,
  `Cname` varchar(50) NOT NULL,
  `Gender` varchar(10) NOT NULL,
  `Age` int NOT NULL,
  `Contact` bigint NOT NULL,
  `Address` varchar(100) NOT NULL)
```

UNIQUE:

This restriction is used to guarantee that each value in a column or collection of columns is distinct. For example, the customer table's contact column needs to be distinct.

```
CREATE TABLE `customer` (
  `CID` int NOT NULL,
  `Cname` varchar(50) NOT NULL,
  `Gender` varchar(10) NOT NULL,
  `Age` int NOT NULL,
  `Contact` bigint NOT NULL,
  `Address` varchar(100) NOT NULL,
  PRIMARY KEY (`CID`),
  UNIQUE KEY `Contact_UNIQUE` (`Contact`);
```

FOREIGN KEY:

This restriction makes sure that the values in one table's column match those in another table's column. For instance, the CID column in the customer table and the invoice table should match.

```
CREATE TABLE `invoice` (
  `invoiceID` int NOT NULL AUTO_INCREMENT,
  `CID` int NOT NULL,
```

```
  `docID` int NOT NULL,
  `medID` int NOT NULL,
  `Quantity` int NOT NULL,
  `Amount` int NOT NULL,
  `PID` int NOT NULL,
  PRIMARY KEY (`invoiceID`),
  CONSTRAINT `CID` FOREIGN KEY (`CID`) REFERENCES `customer` (`CID`),
  CONSTRAINT `docID` FOREIGN KEY (`docID`) REFERENCES `doctor` (`docID`),
  CONSTRAINT `medID` FOREIGN KEY (`medID`) REFERENCES `medicine`
(`medID`));
```

PRIMARY KEY:

A primary key, is a constraint that specifies a column or set of columns that uniquely identifies each row in a database. For instance, the primary key in the employee database should be the EID column.

```
CREATE TABLE `employee` (
  `EID` int NOT NULL AUTO_INCREMENT,
  `Ename` varchar(50) NOT NULL,
  `Gender` varchar(10) NOT NULL,
  `Age` int NOT NULL,
  `Address` varchar(100) NOT NULL,
  `Contact` bigint NOT NULL,
  `Salary` bigint NOT NULL,
  PRIMARY KEY (`EID`));
```

Relational Schema