Welcome you all

MySQL

Day 2

D.Sakthivel
Assistant Professor & Trainer,
KGiSL Micro College
KGiSL Campus, Coimbatore – 641 035.

Day 2 & 3

- Arithmetic Operators
- Between and Not Between
- Date Functions
- Modeling Database
- Applying ER Model Concepts
- ER Model to Relational Database
- Constraint
- Not Null
- Primary Key
- Foreign Key
- Check Constraint
- Set Operation
- JOIN – Natural, Inner Join, Right, Left Join
- JOIN with Group by and Aggregate Function

Arithmetic Operator:

MySQL Arithmetic Operators	**Operation**	**Example**
+	Addition Operator	SELECT 10 + 2 = 12
–	Subtraction Operator	SELECT 10 – 2 = 8
*	Multiplication Operator	SELECT 10 * 2 = 20
/	Division Operator	SELECT 10 / 2 = 5
DIV	Integer Division	SELECT 10 / 2 = 5
% or MOD	Modulus Operator	SELECT 10 % 2 = 0 SELECT 10 % 3 = 1

Arithmetic Operator:

```
mysql> select 10+5, 25+63
    -> ;
+------+-------+
| 10+5 | 25+63 |
+------+-------+
|   15 |    88 |
+------+-------+
1 row in set (0.00 sec)

mysql> select 10-2, 95-58
    -> ;
+------+-------+
| 10-2 | 95-58 |
+------+-------+
|    8 |    37 |
+------+-------+
1 row in set (0.00 sec)
```

```
mysql> select 98 div 2;
+----------+
| 98 div 2 |
+----------+
|       49 |
+----------+
1 row in set (0.00 sec)

mysql> select 98 % 2;
+--------+
| 98 % 2 |
+--------+
|      0 |
+--------+
1 row in set (0.00 sec)
```

```
mysql> select 10-2, 95-58
    -> ;
+------+-------+
| 10-2 | 95-58 |
+------+-------+
|    8 |    37 |
+------+-------+
1 row in set (0.00 sec)

mysql> select 125*5,136*8;
+-------+-------+
| 125*5 | 136*8 |
+-------+-------+
|   625 |  1088 |
+-------+-------+
1 row in set (0.00 sec)

mysql> select 98/2,963/2;
+---------+----------+
| 98/2    | 963/2    |
+---------+----------+
| 49.0000 | 481.5000 |
+---------+----------+
1 row in set (0.00 sec)
```

Arithmetic Operator:

```
mysql> select prodid,prodname,sum(price) from prod_det group by prodname having sum(price)/3 >1000;
+--------+----------+------------+
| prodid | prodname | sum(price) |
+--------+----------+------------+
|    101 | samsung  |   45000.75 |
|    102 | HP       |  150000.78 |
|    103 | DELL     |   35000.65 |
|    105 | AUZ      |   60000.26 |
+--------+----------+------------+
4 rows in set (0.00 sec)
```

Between and Not Between:

```
mysql> select * from prod_det where price not between 15000 and 50000;
+--------+----------+----------+
| prodid | prodname | price    |
+--------+----------+----------+
|    102 | HP       | 50000.26 |
|    102 | HP       | 50000.26 |
|    102 | HP       | 50000.26 |
|    105 | AUZ      | 60000.26 |
+--------+----------+----------+
4 rows in set (0.00 sec)

mysql> select * from prod_det where price  between 15000 and 50000;
+--------+----------+----------+
| prodid | prodname | price    |
+--------+----------+----------+
|    101 | samsung  | 15000.25 |
|    101 | samsung  | 15000.25 |
|    101 | samsung  | 15000.25 |
|    103 | DELL     | 35000.65 |
+--------+----------+----------+
4 rows in set (0.00 sec)
```

MySQL Date Functions:

- ✓ Along with strings and numbers, you often need to store date and/or time values in a database, such as an user's birth date, employee's hiring date, date of the future events, the date and time a particular row is created or modified in a table, and so on.
- ✓ This type of data is referred as temporal data and every database engine has a default storage format and data types for storing them.

KGiSL MicroCollege

MySQL Date Functions:

Type	Default format	Allowable values
DATE	YYYY-MM-DD	1000-01-01 to 9999-12-31
TIME	HH:MM:SS or HHH:MM:SS	-838:59:59 to 838:59:59
DATETIME	YYYY-MM-DD HH:MM:SS	1000-01-01 00:00:00 to 9999-12-31 23:59:59
TIMESTAMP	YYYY-MM-DD HH:MM:SS	1970-01-01 00:00:00 to 2037-12-31 23:59:59
YEAR	YYYY	1901 to 2155

MySQL Date Functions:

MySQL Curdate is a Date Function, which is useful to return the current date value in YYYY-MM-DD or YYYYMMDD format.

MySQL Date Functions	Description
ADDDATE()	Add Date Value in intervals to a given Date.
ADDTIME()	This adds Time
CONVERT_TZ()	This MySQL Date function convert date and time from one time zone to another.
CURDATE()	Returns the Current date
CURRENT_DATE, CURRENT_DATE()	Synonym of CURDATE()
CURRENT_TIME(), CURRENT_TIME	Synonym of CURTIME() function
CURRENT_TIMESTAMP() , CURRENT_TIMESTAMP	Synonym of NOW() function
CURTIME()	This MySQL function returns the current Time
DATE()	It extracts the date part from the Given Date or DateTime expression

MySQL Date Functions:

DATE_ADD()	It adds a given intervals to the date expression
DATE_FORMAT()	This method formats the Date as per our requirements.
DATE_SUB()	This subtracts the specified intervals from the given date.
DATEDIFF()	Subtracts one date from another and returns the difference.
DAY()	Synonym of DAYOFMONTH() function.
DAYNAME()	This MySQL date function returns the name of the Weekday. For example Monday
DAYOFMONTH()	It returns the day number of the Month from 1 to 31
DAYOFWEEK()	This returns the Weekday index of the user given date.
DAYOFYEAR()	Returns the Day of the Year or Day number, i.e., 1 to 366
EXTRACT()	Use this to extract part of a date.
FROM_DAYS()	This MySQL date function converts given day number to date.

MySQL Date Functions:

GET_FORMAT()	This returns the date format string.
HOUR()	Returns the Hour value from a given Time.
LAST_DAY	It returns the last Day of the Month on a given date.
LOCALTIME(), LOCALTIME	Synonym of NOW() function
LOCALTIMESTAMP(), LOCALTIMESTAMP	Synonym of NOW() function
MAKEDATE()	It is used to create or make a date from the specified Year, and day of the year
MAKETIME()	This will Create or make time from Hour, Minute, Second
MICROSECOND()	Returns the Microseconds from the given Time or DateTime.
MINUTE()	This function returns the Minutes value from given Time or DateTime.
MONTH()	It retunes the Month Number (1 – 12)
MONTHNAME()	Returns the Month name or Name of the Month (January, February, etc.)
NOW()	This MySQL Date function returns the Current Date and Time.

KGiSL MicroCollege

MySQL Date Functions:

PERIOD_ADD()	It adds the user specified period to a year-Month
PERIOD_DIFF()	Returns the difference between the two periods. It returns the number of months between those two periods.
SECOND()	Returns the seconds value (0-59) from a given Time.
STR_TO_DATE()	This function converts String to date.
SYSDATE()	Returns the current system date and time at which the function executed
TIME()	This MySQL date function extracts the time portion from given DateTime or expression
TIME_FORMAT()	Use this to format the given expression as time.
TIME_TO_SEC()	This method converts the given time to second.
TIMEDIFF()	It subtracts time.
TIMESTAMP()	It returns the Date or DateTime expression.

MySQL Date Functions:

ADDDATE(Date, INTERVAL expression Unit);

```
mysql> SELECT ADDDATE('2022-02-28 23:59:59', INTERVAL 31 DAY);
+-----------------------------------------------+
| ADDDATE('2022-02-28 23:59:59', INTERVAL 31 DAY) |
+-----------------------------------------------+
| 2022-03-31 23:59:59                           |
+-----------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT ADDDATE('2022-02-28 23:59:59', INTERVAL 16 WEEK);
+-----------------------------------------------+
| ADDDATE('2022-02-28 23:59:59', INTERVAL 16 WEEK) |
+-----------------------------------------------+
| 2022-06-20 23:59:59                           |
+-----------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT ADDDATE('2022-02-28 23:59:59', INTERVAL 18 MONTH);
+-----------------------------------------------+
| ADDDATE('2022-02-28 23:59:59', INTERVAL 18 MONTH) |
+-----------------------------------------------+
| 2023-08-28 23:59:59                           |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

MySQL Date Functions:

ADDTIME(DateTime1 or Time_Expression1, Time_Expression2);

```
mysql> SELECT ADDTIME('10:11:22', '12:10:12');
+----------------------------------+
| ADDTIME('10:11:22', '12:10:12') |
+----------------------------------+
| 22:21:34                         |
+----------------------------------+
1 row in set (0.00 sec)

mysql> SELECT ADDTIME('10:11:22', '12:10:12.111222');
+----------------------------------------+
| ADDTIME('10:11:22', '12:10:12.111222') |
+----------------------------------------+
| 22:21:34.111222                        |
+----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT ADDTIME('10:11:22.333444', '12:10:12.111222');
+-----------------------------------------------+
| ADDTIME('10:11:22.333444', '12:10:12.111222') |
+-----------------------------------------------+
| 22:21:34.444666                               |
+-----------------------------------------------+
1 row in set (0.00 sec)
```

MySQL Date Functions:

CONVERT_TZ(Date, from_timezone, to_timezone):

```
mysql> SELECT CONVERT_TZ('2022-02-28 23:59:59', '+00:00', '+05:50');
+-----------------------------------------------------+
| CONVERT_TZ('2022-02-28 23:59:59', '+00:00', '+05:50') |
+-----------------------------------------------------+
| 2022-03-01 05:49:59                                 |
+-----------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT CONVERT_TZ('2019-02-28 23:59:59', '+00:00', '-05:50');
+-----------------------------------------------------+
| CONVERT_TZ('2019-02-28 23:59:59', '+00:00', '-05:50') |
+-----------------------------------------------------+
| 2019-02-28 18:09:59                                 |
+-----------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT CONVERT_TZ('2019-02-28 23:59:59', '+00:00', '+10:30');
+-----------------------------------------------------+
| CONVERT_TZ('2019-02-28 23:59:59', '+00:00', '+10:30') |
+-----------------------------------------------------+
| 2019-03-01 10:29:59                                 |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

MySQL Date Functions:

CURRENT DATE,TIME,HOURS,MINUTES,SECONDS:

```
mysql> SELECT CURDATE();
+-------------+
| CURDATE()   |
+-------------+
| 2022-07-28  |
+-------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT CURRENT_TIMESTAMP();
+---------------------+
| CURRENT_TIMESTAMP() |
+---------------------+
| 2022-07-28 13:43:54 |
+---------------------+
1 row in set (0.00 sec)

mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2022-07-28 13:44:04 |
+---------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT CURRENT_TIME();
+----------------+
| CURRENT_TIME() |
+----------------+
| 13:43:13       |
+----------------+
1 row in set (0.00 sec)
```

MySQL DATEDIFF Function

MySQL DATEDIFF function is one of the Date methods, which is useful to find the difference between two dates and returns the number of days.

DATEDIFF(Expression1, Expression2);

```
mysql> SELECT DATEDIFF('2021-02-28', '2020-01-01');
+---------------------------------------+
| DATEDIFF('2021-02-28', '2020-01-01') |
+---------------------------------------+
|                                   424 |
+---------------------------------------+
1 row in set (0.00 sec)
```

MySQL Day Function

MySQL Day is one of the Date Functions, which is useful to return the day of the month range from 0 to 31.

Or, this MySQL function returns the Day number from a given date or DateTime expression.

sion);

```
mysql> SELECT DAY('2016-11-25');
+-------------------+
| DAY('2016-11-25') |
+-------------------+
|                25 |
+-------------------+
1 row in set (0.00 sec)

mysql> SELECT DAY('2018-10-25 01:14:22');
+----------------------------+
| DAY('2018-10-25 01:14:22') |
+----------------------------+
|                         25 |
+----------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT DATE_FORMAT('2018-12-31 23:59:02', '%Y');
+-----------------------------------------+
| DATE_FORMAT('2018-12-31 23:59:02', '%Y') |
+-----------------------------------------+
| 2018                                    |
+-----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_FORMAT('2018-12-31 23:59:02', '%p');
+-----------------------------------------+
| DATE_FORMAT('2018-12-31 23:59:02', '%p') |
+-----------------------------------------+
| PM                                      |
+-----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_FORMAT('2018-12-31 23:59:02', '%a');
+-----------------------------------------+
| DATE_FORMAT('2018-12-31 23:59:02', '%a') |
+-----------------------------------------+
| Mon                                     |
+-----------------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT DATE_FORMAT('2018-12-31 23:59:02', '%W');
+-----------------------------------------+
| DATE_FORMAT('2018-12-31 23:59:02', '%W') |
+-----------------------------------------+
| Monday                                  |
+-----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_FORMAT('2018-12-31 23:59:02', '%U');
+-----------------------------------------+
| DATE_FORMAT('2018-12-31 23:59:02', '%U') |
+-----------------------------------------+
| 52                                      |
+-----------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT DATE_FORMAT('2018-12-31 23:59:02', '%u');
+-----------------------------------------+
| DATE_FORMAT('2018-12-31 23:59:02', '%u') |
+-----------------------------------------+
| 53                                      |
+-----------------------------------------+
1 row in set (0.00 sec)
```

MySQL Time Function

MySQL Time Function is useful to extract the Time part from the given DateTime. In this study, we show you how to use this method to extract the time value from the given expression with an example and the syntax is

TIME(Time or DateTime or expression);

```
mysql> SELECT TIME('10:12:34');
+-------------------+
| TIME('10:12:34') |
+-------------------+
| 10:12:34          |
+-------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT SECOND('10:22:45');
+--------------------+
| SECOND('10:22:45') |
+--------------------+
|                 45 |
+--------------------+
1 row in set (0.00 sec)

mysql>
mysql> SELECT MINUTE('12:35:44');
+--------------------+
| MINUTE('12:35:44') |
+--------------------+
|                 35 |
+--------------------+
1 row in set (0.00 sec)
```

Modelling Databases: Part 1

- To model a database, we have to first understand the business requirements at conceptual level, which is later translated into a relational database.
- For understanding the business requirements at a conceptual level, we use Entity Relationship Model (ER Model)

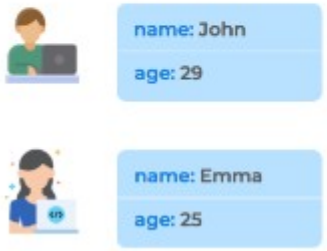
Core Concepts in ER Model

Entity

Real world objects/concepts are called **entities** in ER Model.

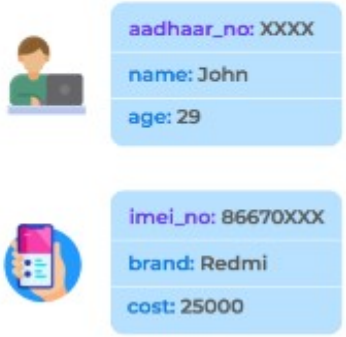
Modelling Databases: Part 1

Attributes of an Entity



Properties of real world objects/concepts are represented as **attributes** of an entity in ER model.

Key Attribute



The attribute that uniquely identifies each entity is called **key attribute**.

Modelling Databases: Part 1

Entity Type

Entity Type is a **collection of entities** that have the same attributes (not values).

Modelling Databases: Part 1

Relationships

Association among the entities is called a **relationship**.

Example:

Person **has a** passport.
Person can **have many** cars.

Each student can **register for many** courses, and a course can **have many** students.

Types of relationships

1. **One-to-One Relationship**
2. **One-to-Many or Many-to-One Relationship**
3. **Many-to-Many Relationship**

Modelling Databases: Part 1

One-to-One Relationship

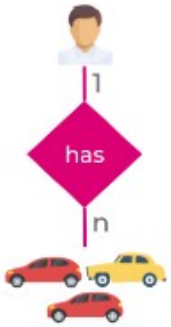
An entity is related to **only one entity**, and vice versa.

Example

- A person can have **only one** passport.
- similarly, a passport belongs to **one and only one** person.

Modelling Databases: Part 1

One-to-Many Relationship

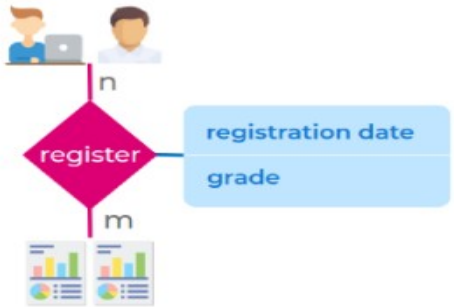


An entity is related to **many other** entities.

Example

- A person **can have many** cars. But a car belongs to **only one** person.

Many-to-Many Relationship



Multiple entities are related to multiple entities.

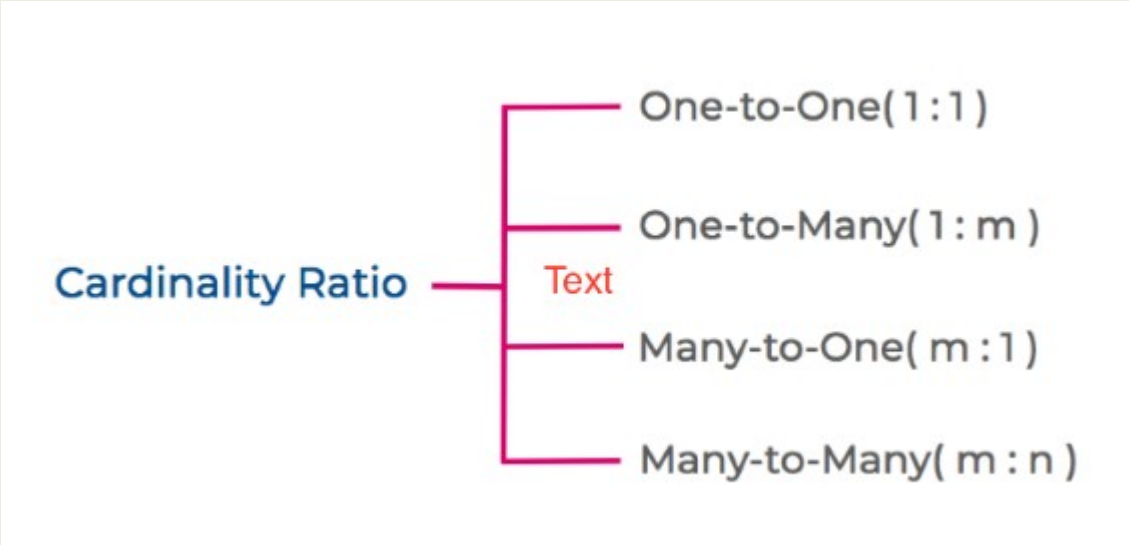
Example

- Each student can register to multiple courses.
- similarly each course is taken by multiple students.

Modelling Databases: Part 1

Cardinality Ratio

Cardinality in DBMS defines the maximum number of times an instance in one entity can relate to instances of another entity.



Applying ER Model Concepts

Applying ER Model Concepts

E-commerce Application

In a typical e-commerce application,

- *Customer* has only one *cart*. A *cart* belongs to only one *customer*
- *Customer* can add *products* to *cart*
- *Cart* contains multiple *products*
- *Customer* can save multiple *addresses* in the application for further use like selecting delivery address

Let's apply the concepts of ER Model to this e-commerce scenario.

Entity types

- Customer
- Product
- Cart
- Address

Relationships

Applying ER Model Concepts

Relation Between Cart and Customer

- A customer has **only one** cart.
- A cart is related to **only one** customer.
- Hence, the relation between customer and cart entities is **One-to-One relation**.

Applying ER Model Concepts

Relation Between Cart and Products

- A cart can have **many** products.
- A product can be in **many** carts.
- Therefore, the relation between cart and product is **Many-to-Many relation**.

Applying ER Model Concepts

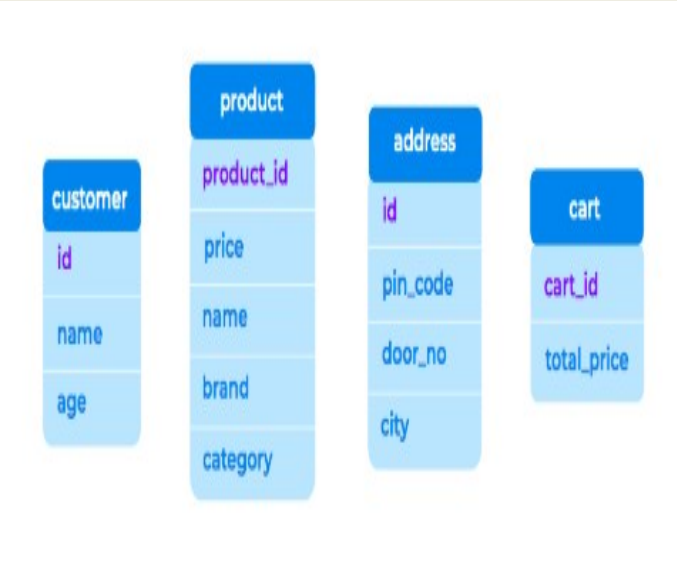
Relation Between Customer and Address

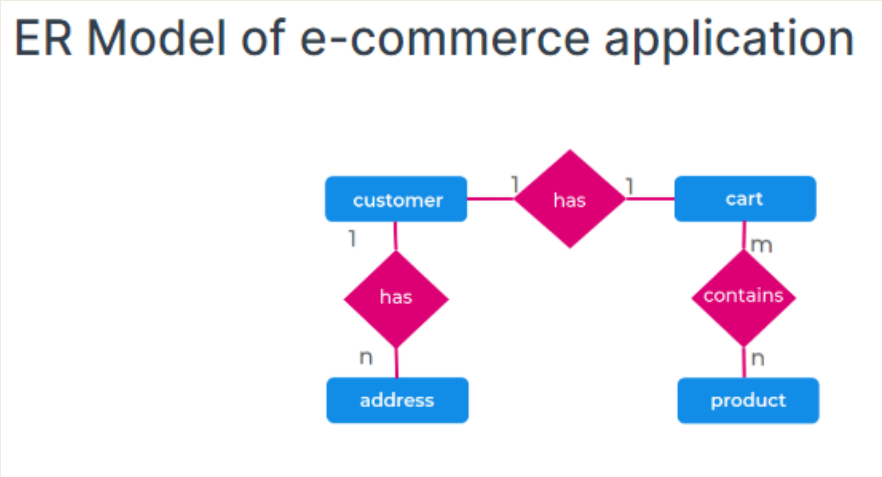
- A customer can have **multiple** addresses.
- An address is related to **only one** customer.
- Hence, the relation between customer and address is **One-to-Many relation**.

Applying ER Model Concepts

Attributes

Following are the attributes for the entity types in the e-commerce scenario. Here, attributes like id, product_id, etc., are **key attributes** as they **uniquely identify each entity** in the entity type.



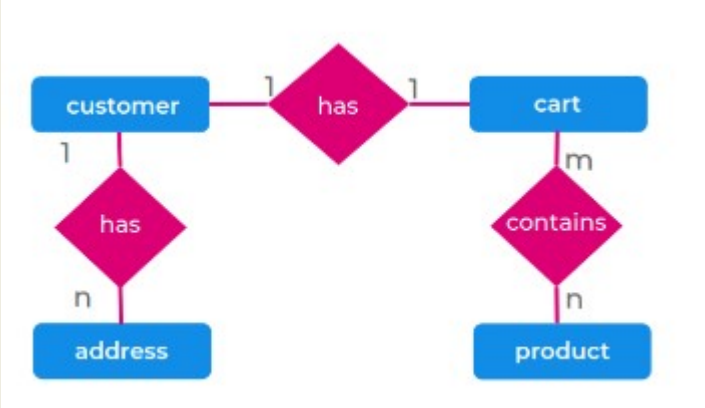


ER Model to Relational Database

E-commerce Application

In a typical e-commerce application,

- *Customer* has only one *cart*. A *cart* belongs to only one *customer*
- *Customer* can add products to *cart*
- *Cart* contains multiple *products*
- *Customer* can save multiple *addresses* in the application for further use like selecting delivery address



ER Model to Relational Database

Entity Type to Table

Entity Types → Tables

Attributes → Columns

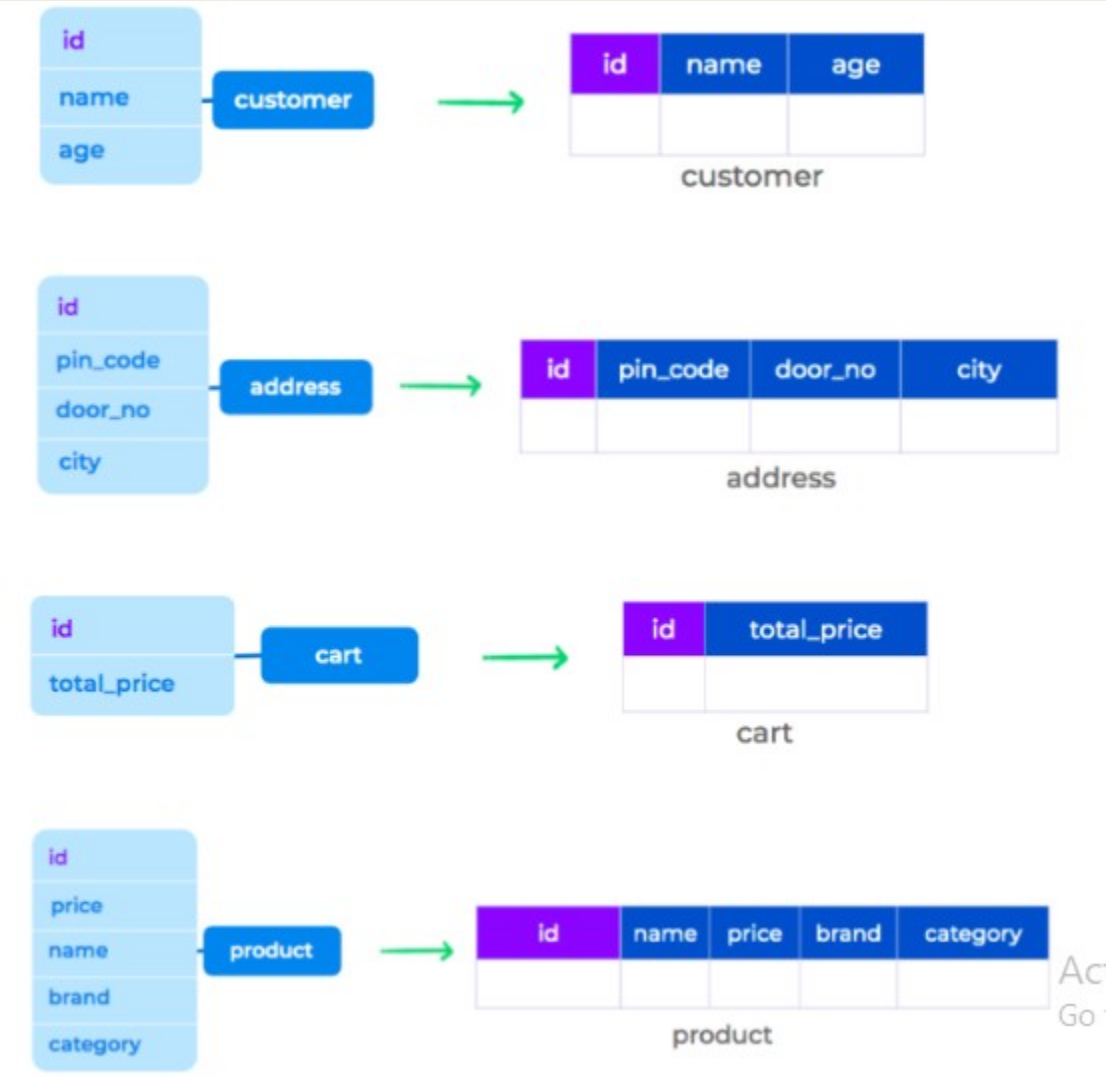
Key Attribute → Primary Key

Primary key: A minimal set of attributes (columns) in a table that uniquely identifies rows in a table.

In the following tables, *all the ids are primary keys* as they *uniquely identify each row* in the table.

KGiSL MicroCollege

ER Model to Relational Database



ER Model to Relational Database

Relationships

Relation Between Customer and Address - One to Many Relationship

A customer can have multiple addresses.
An address is related to only one customer.
We store the primary key of a customer in the address table to denote that the addresses are related to a particular customer. This new column/s in the table that refer to the primary key

id	pin_code	door_no	...	customer_id
1001	517130	6-1	...	1
1002	615670	6-13	...	1

address

PK
FK
Unique FK

Here, customer_id is the foreign key that stores id (primary key) of customers.

ER Model to Relational Database

Relation Between Cart and Customer - One to One Relationship

- A customer has only one cart.
- A cart is related to only one customer.

This is similar to one-to-many relationship. But, we need to ensure that *only one cart is associated to a customer*

id	total_price	customer_id
1	1200	1
2	500	2

→ FK

cart

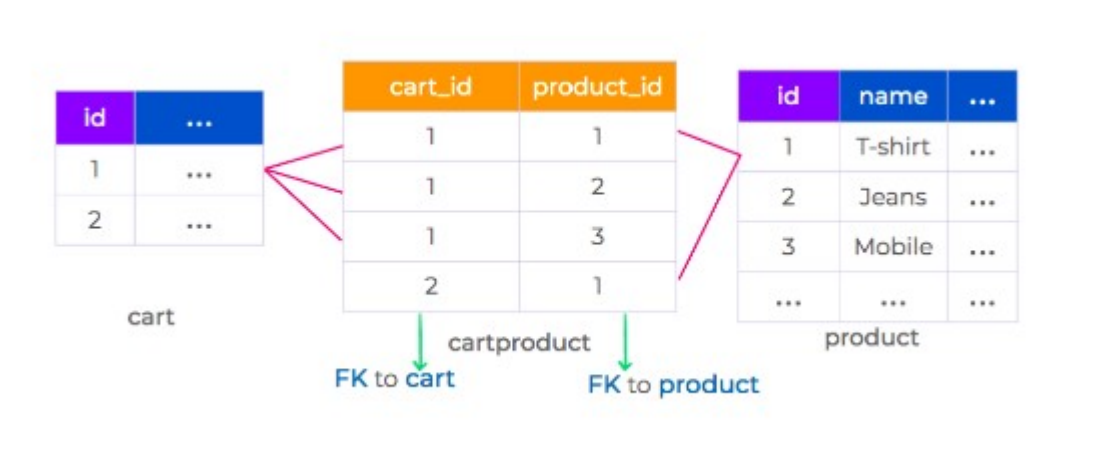
■ → PK
■ → FK
■ → Unique FK

ER Model to Relational Database

Relation Between Cart and Products - Many to Many Relationship

- A cart can have many products.
- A product can be in many carts.

Here, we cannot store either the primary key of a product in the cart table or vice versa. To store the relationship between the cart and product tables, we use a **Junction Table**.



Creating a Relational Database

In the previous sessions, we've explored how to represent an ER model in the form of tables in a relational database.

Now, let's create tables to store the data in the database by defining all the columns and relationships between the tables.

Consider the **e-commerce scenario**. The tables, columns and the relations between them are s follows.

PK
FK
Unique FK

KGiSL MicroCollege

id	name	age

customer

id	pin_code	door_no	city	customer_id

address

id	total_price	customer_id

cart

id	name	price	brand	category

product

id	cart_id	product_id	quantity

cartproduct

KGiSL MicroCollege

What is Constraint?

A constraint is simply a restriction placed on one or more columns of a table to limit the type of values that can be stored in that column.

Constraints provide a standard mechanism to maintain the accuracy and integrity of the data inside a database table.

There are several different types of constraints in SQL, including:

- NOT NULL
- PRIMARY KEY
- FOREIGN KEY
- CHECK

Null Values

```
mysql> create table demo_null(id int);
Query OK, 0 rows affected (0.08 sec)

mysql> insert into demo_null values(10);
Query OK, 1 row affected (0.05 sec)

mysql> insert into demo_null values(12);
Query OK, 1 row affected (0.00 sec)

mysql> insert into demo_null values(99);
Query OK, 1 row affected (0.00 sec)

mysql> insert into demo_null values( );
Query OK, 1 row affected (0.00 sec)

mysql> select * from demo_null;
+------+
| id   |
+------+
|   10 |
|   12 |
|   99 |
| NULL |
+------+
4 rows in set (0.00 sec)
```

NOT NULL Constraint

The NOT NULL constraint specifies that the column does not accept NULL values.

This means if NOT NULL constraint is applied on a column then you cannot insert a new row in the table without adding a non-NULL value for that column.

```
mysql> create table demo_NotNull(sno int NOT NULL,name varchar(20));
Query OK, 0 rows affected (0.10 sec)

mysql> insert into demp_NotNull(sno,name) values(101,Raj);
ERROR 1146 (42S02): Table 'student_db.demp_notnull' doesn't exist
mysql> insert into demp_NotNull(sno,name) values(101,'Raj');
ERROR 1146 (42S02): Table 'student_db.demp_notnull' doesn't exist
mysql> insert into demo_NotNull(sno,name) values(101,'Raj');
Query OK, 1 row affected (0.00 sec)

mysql> insert into demo_NotNull(sno,name) values(102,'Sekar');
Query OK, 1 row affected (0.00 sec)

mysql> insert into demo_NotNull(sno,name) values(,'Murali');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
 for the right syntax to use near ','Murali')' at line 1
```

Primary Key

Following syntax creates a table with `c1` as the primary key.

Syntax

```
CREATE TABLE table_name (
    c1 t1 NOT NULL PRIMARY KEY,
...
    cn tn,
);
```

Foreign Key

In case of foreign key, we just create a foreign key constraint.

Syntax

```sql
CREATE TABLE table2(
  c1 t1 NOT NULL PRIMARY KEY,
  c2 t2,
  FOREIGN KEY(c2) REFERENCES table1(c3) ON DELETE CASCADE
);
```

Understanding

SQL

```sql
FOREIGN KEY(c2) REFERENCES table1(c3)
```

Above part of the foreign key constraint ensure that foreign key can only contain values that are in the referenced primary key.

SQL

```sql
ON DELETE CASCADE
```

Ensure that if a row in `table1` is deleted, then all its related rows in `table2` will also be deleted.

KGiSL MicroCollege

Primary Key:

- A primary key is a column or a set of columns that uniquely identifies each row in the table.

The primary key follows these rules:

- A primary key must contain unique values.
- If the primary key consists of multiple columns, the combination of values in these columns must be unique.
- A primary key column cannot have NULL values.
- Any attempt to insert or update NULL to primary key columns will result in an error.
- Note that MySQL implicitly adds a NOT NULL constraint to primary key columns.
- A table can have one an only one primary key.

A primary key column often has the AUTO_INCREMENT attribute that automatically generates a sequential integer whenever you insert a new row into the table.

KGiSL MicroCollege

Primary Key:
Syntax:

```
CREATE TABLE table_name(
    primary_key_column datatype PRIMARY KEY,
    ...
);
```

```
CREATE TABLE table_name(
    primary_key_column1 datatype,
    primary_key_column2 datatype,
    ...,
    PRIMARY KEY(column_list)
);
```

Primary Key- Example:

```
mysql> select * from prod_det;
+--------+----------+----------+
| prodid | prodname | price    |
+--------+----------+----------+
|    101 | samsung  | 15000.25 |
|    102 | HP       | 50000.26 |
|    101 | samsung  | 15000.25 |
|    102 | HP       | 50000.26 |
|    101 | samsung  | 15000.25 |
|    102 | HP       | 50000.26 |
|    103 | DELL     | 35000.65 |
|    105 | AUZ      | 60000.26 |
+--------+----------+----------+
8 rows in set (0.00 sec)

mysql> create table product_det(prodid int primary key, prodname varchar(25), price double(10,2));
Query OK, 0 rows affected, 1 warning (0.11 sec)
```

```
mysql> desc product_det;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| prodid   | int          | NO   | PRI | NULL    |       |
| prodname | varchar(25)  | YES  |     | NULL    |       |
| price    | double(10,2) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.05 sec)
```

Primary Key- Example:

```
mysql> insert into product_det(prodid,prodname,price) values(101,'samsung',15000.25),(102,'HP',50000.25);
Query OK, 2 rows affected (0.04 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> insert into product_det(prodid,prodname,price) values(103,'samsungAX',25000.75),(104,'DELL',35000.25),(105,'ASUS'
,28000.63);
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from product_det;
+--------+-----------+----------+
| prodid | prodname  | price    |
+--------+-----------+----------+
|    101 | samsung   | 15000.25 |
|    102 | HP        | 50000.25 |
|    103 | samsungAX | 25000.75 |
|    104 | DELL      | 35000.25 |
|    105 | ASUS      | 28000.63 |
+--------+-----------+----------+
5 rows in set (0.00 sec)
```

Primary Key- Example:

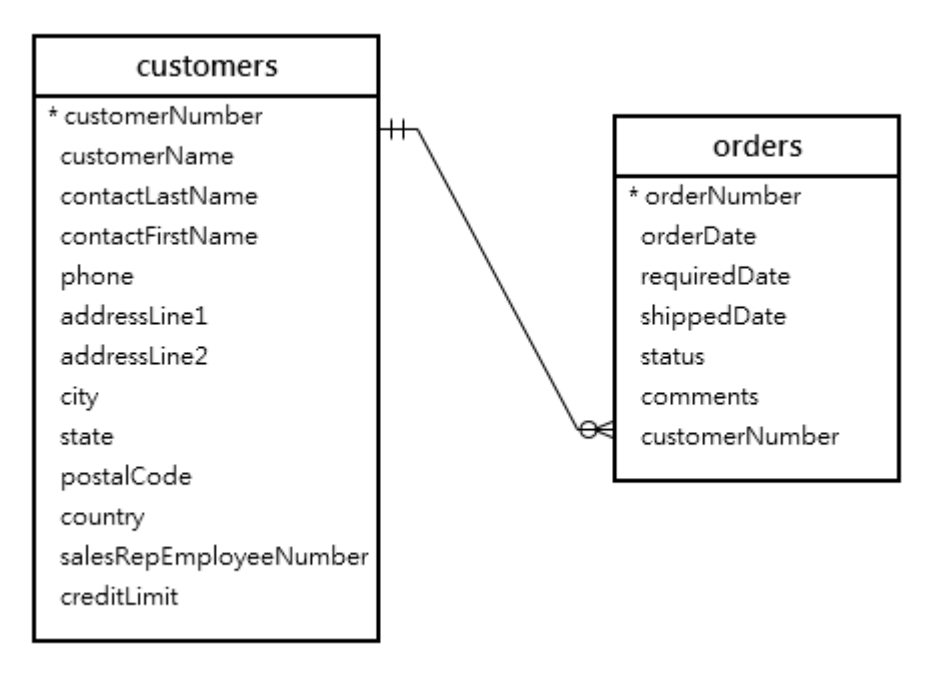
```
mysql> insert into product_det values(101,'samsungDX',36000.90);
ERROR 1062 (23000): Duplicate entry '101' for key 'product_det.PRIMARY'
```

```
mysql> insert into product_det values(106,'samsungDX',36000.90);
Query OK, 1 row affected (0.04 sec)

mysql> select * from product_det;
+--------+-----------+----------+
| prodid | prodname  | price    |
+--------+-----------+----------+
|    101 | samsung   | 15000.25 |
|    102 | HP        | 50000.25 |
|    103 | samsungAX | 25000.75 |
|    104 | DELL      | 35000.25 |
|    105 | ASUS      | 28000.63 |
|    106 | samsungDX | 36000.90 |
+--------+-----------+----------+
6 rows in set (0.00 sec)
```

Foreign Key:

- A foreign key is a column or group of columns in a table that links to a column or group of columns in another table.
- The foreign key places constraints on data in the related tables, which allows MySQL to maintain referential integrity.



Foreign Key:

- **Syntax:**
- **CREATE TABLE** table_name (**FOREIGN KEY** (column_name) **REFERENCE** table_name(Referencing column_name in table_name));

```
mysql> create table cust_det(custid int primary key, custname varchar(25),prodid int,
 foreign key(prodid) REFERENCES product_det(prodid));
Query OK, 0 rows affected (0.06 sec)
```

Foreign Key:

```
mysql> insert into cust_det(custid,custname,prodid) values(001,'Abijith',101);
Query OK, 1 row affected (0.01 sec)

mysql> insert into cust_det(custid,custname,prodid) values(003,'Murali',103);
Query OK, 1 row affected (0.04 sec)
```

```
mysql> insert into cust_det(custid,custname,prodid) values(004,'Raghu',101),(005,'Rek
ha',104),(010,'Hari',105);
Query OK, 3 rows affected (0.05 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from cust_det;
+--------+----------+--------+
| custid | custname | prodid |
+--------+----------+--------+
|      1 | Abijith  |    101 |
|      3 | Murali   |    103 |
|      4 | Raghu    |    101 |
|      5 | Rekha    |    104 |
|     10 | Hari     |    105 |
+--------+----------+--------+
5 rows in set (0.00 sec)
```

```
mysql> insert into cust_det values(11,'John',120);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
(`student_db`.`cust_det`, CONSTRAINT `cust_det_ibfk_1` FOREIGN KEY (`prodid`) REFEREN
CES `product_det` (`prodid`))
```

MySQL AUTO_INCREMENT Keyword:

- Auto-increment allows a **unique number to be generated automatically** when a new record is inserted into a table.
- Often this is the primary key field that we would like to be created automatically every time a new record is inserted.
- MySQL uses the AUTO_INCREMENT keyword to perform an auto-increment feature.
- By default, the starting value for AUTO_INCREMENT is 1, and it will increment by 1 for each new record.

MySQL AUTO_INCREMENT Keyword:

```
mysql> create table student_info(sno int NOT NULL AUTO_INCREMENT,sname varchar(24),age int, address varchar(30),PRIMARY
KEY(sno));
Query OK, 0 rows affected (0.14 sec)
```

```
mysql> insert into student_info(sname,age,address)values('Arunkumar',21,'Madurai'),('Divya',19,'Coimbatore'),('Farooq',1
8,'Salem'),('Mani',20,'Coimbatore');
Query OK, 4 rows affected (0.05 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from student_info;
+-----+-----------+------+------------+
| sno | sname     | age  | address    |
+-----+-----------+------+------------+
|   1 | Arunkumar |   21 | Madurai    |
|   2 | Divya     |   19 | Coimbatore |
|   3 | Farooq    |   18 | Salem      |
|   4 | Mani      |   20 | Coimbatore |
+-----+-----------+------+------------+
4 rows in set (0.00 sec)
```

MySQL AUTO_INCREMENT Keyword:

To let the AUTO_INCREMENT sequence start with another value, use the following SQL statement:

ALTER TABLE student_info

```
mysql> alter table student_info AUTO_INCREMENT =100;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> insert into student_info(sname,age,address)values('Ganesh',21,'Madurai'),('Kalaivani',19,'Coimbatore');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from student_info;
+-----+-----------+------+------------+
| sno | sname     | age  | address    |
+-----+-----------+------+------------+
|   1 | Arunkumar |   21 | Madurai    |
|   2 | Divya     |   19 | Coimbatore |
|   3 | Farooq    |   18 | Salem      |
|   4 | Mani      |   20 | Coimbatore |
| 100 | Ganesh    |   21 | Madurai    |
| 101 | Kalaivani |   19 | Coimbatore |
+-----+-----------+------+------------+
6 rows in set (0.00 sec)
```

MySQL AUTO_INCREMENT Keyword:

```
mysql> create table cust_det(custid int NOT NULL AUTO_INCREMENT,custname varchar(20),PRIMARY KEY(custid)) auto_increment=2000;
Query OK, 0 rows affected (0.07 sec)

mysql> insert into cust_det(custname) values('Arun'),('Babu'),('Chandra'),('Madura');
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from cust_det;
+--------+----------+
| custid | custname |
+--------+----------+
|   2000 | Arun     |
|   2001 | Babu     |
|   2002 | Chandra  |
|   2003 | Madura   |
+--------+----------+
4 rows in set (0.00 sec)
```

CHECK Constraint

The CHECK constraint is used to restrict the values that can be placed in a column.

For example, the range of values for a price column can be limited by creating
a CHECK constraint that allows values only from 3,000 to 10,000. This prevents price from

```
mysql> create table demo_check(prodid int,prodname varchar(20),price double(10,2) check(price >=3000 and price<=10000));

Query OK, 0 rows affected, 1 warning (0.06 sec)

mysql> desc demo_check;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| prodid   | int          | YES  |     | NULL    |       |
| prodname | varchar(20)  | YES  |     | NULL    |       |
| price    | double(10,2) | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
3 rows in set (0.05 sec)
```

CHECK Constraint

```
mysql> insert into demo_check(prodid,prodname,price) values(101,'mp3 player',4356.50),(145,'Doorbell camera',8999.99);
Query OK, 2 rows affected (0.04 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from demo_check;
+--------+----------------+---------+
| prodid | prodname       | price   |
+--------+----------------+---------+
|    101 | mp3 player     | 4356.50 |
|    145 | Doorbell camera | 8999.99 |
+--------+----------------+---------+
2 rows in set (0.00 sec)

mysql> insert into demo_check values(589,'Projector',12000.99);
ERROR 3819 (HY000): Check constraint 'demo_check_chk_1' is violated.
```

Set Operation:

Union:

MySQL Union is an operator that allows us to combine two or more results from multiple SELECT queries into a single result set.

It comes with a default feature that removes the **duplicate** rows from the result set.

```
SELECT column_list FROM table1
UNION
SELECT column_list FROM table2;
```

Set Operation:

Union:

```
mysql> select * from product_det;
+--------+-----------+----------+
| prodid | prodname  | price    |
+--------+-----------+----------+
|    101 | samsung   | 15000.25 |
|    102 | HP        | 50000.25 |
|    103 | samsungAX | 25000.75 |
|    104 | DELL      | 35000.25 |
|    105 | ASUS      | 28000.63 |
|    106 | samsungDX | 36000.90 |
+--------+-----------+----------+
6 rows in set (0.00 sec)

mysql> select * from cust_det;
+--------+----------+--------+
| custid | custname | prodid |
+--------+----------+--------+
|      1 | Abijith  |    101 |
|      3 | Murali   |    103 |
|      4 | Raghu    |    101 |
|      5 | Rekha    |    104 |
|     10 | Hari     |    105 |
+--------+----------+--------+
5 rows in set (0.00 sec)
```

```
mysql> select prodid from product_det UNION select prodid from cust_det;
+--------+
| prodid |
+--------+
|    101 |
|    102 |
|    103 |
|    104 |
|    105 |
|    106 |
+--------+
6 rows in set (0.00 sec)
```

Set Operation:

Union ALL:

MySQL Union is an operator that allows us to combine two or more results from multiple SELECT queries into a single result set.

It comes with a default feature from the result set.

```
SELECT column_list FROM table1
UNION  ALL
SELECT column_list FROM table2;
```

Set Operation:
Union ALL:

```
mysql> select prodid from product_det UNION ALL select prodid from cust_det;
+--------+
| prodid |
+--------+
|    101 |
|    102 |
|    103 |
|    104 |
|    105 |
|    106 |
|    101 |
|    101 |
|    103 |
|    104 |
|    105 |
+--------+
11 rows in set (0.00 sec)
```

Set Operation:

Union ALL:

MySQL Union is an operator that allows us to combine two or more results from multiple SELECT queries into a single result set.

It comes with a default feature from the result set.

```
SELECT column_list FROM table1
UNION  ALL
SELECT column_list FROM table2;
```

JOINS

So far, we have learnt to analyse the data that is present in a single table.

But in the real-world scenarios, often, the data is distributed in multiple tables.

To fetch meaningful insights, we have to bring the data together by combining the tables.

We use JOIN clause to combine rows from two or more tables, based on a related column between them.

There are various types of joins, namely **Natural join, Inner Join, Full Join, Cross Join, Left join, Right join.**

NATURAL JOIN

combines the tables based on the common columns.

Combining two tables:

```
SELECT * FROM table1 INNER JOIN
table2 ON table1.id = table2.id;
```

Combining three tables:

```
SELECT * FROM table1 INNER JOIN
table2 ON table1.id = table2.id
INNER JOIN table3 ON table2.id =
table3.id;
```

```
mysql> create table bill(custid int,prodid int,billamount double(10,2),billdate date,
foreign key(custid) references cust_det(custid), foreign key(prodid) references produ
ct_det(prodid));
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> desc bill;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| custid     | int          | YES  | MUL | NULL    |       |
| prodid     | int          | YES  | MUL | NULL    |       |
| billamount | double(10,2) | YES  |     | NULL    |       |
| billdate   | date         | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
4 rows in set (0.03 sec)
```

```
mysql> insert into bill(custid,prodid,billamount,billdate) values(4, 101,16000.85,'20
22-07-28'),(3,103,26000.45,'2022-06-15'),(3,101,15500,'2022-05-22'),(3,102,52000.75,'
2022-07-14');
Query OK, 4 rows affected (0.05 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from billdate;
ERROR 1146 (42S02): Table 'student_db.billdate' doesn't exist
mysql> select * from bill;
+--------+--------+------------+------------+
| custid | prodid | billamount | billdate   |
+--------+--------+------------+------------+
|      1 |    101 |   16000.85 | 2022-07-28 |
|      4 |    101 |   16000.85 | 2022-07-28 |
|      3 |    103 |   26000.45 | 2022-06-15 |
|      3 |    101 |   15500.00 | 2022-05-22 |
|      3 |    102 |   52000.75 | 2022-07-14 |
+--------+--------+------------+------------+
5 rows in set (0.00 sec)
```

```
mysql> select * from product_det;
+--------+-----------+----------+
| prodid | prodname  | price    |
+--------+-----------+----------+
|    101 | samsung   | 15000.25 |
|    102 | HP        | 50000.25 |
|    103 | samsungAX | 25000.75 |
|    104 | DELL      | 35000.25 |
|    105 | ASUS      | 28000.63 |
|    106 | samsungDX | 36000.90 |
+--------+-----------+----------+
6 rows in set (0.00 sec)

mysql> select * from cust_det;
+--------+----------+--------+
| custid | custname | prodid |
+--------+----------+--------+
|      1 | Abijith  |    101 |
|      3 | Murali   |    103 |
|      4 | Raghu    |    101 |
|      5 | Rekha    |    104 |
|     10 | Hari     |    105 |
|   1235 | Yasin    |    103 |
+--------+----------+--------+
6 rows in set (0.00 sec)
```

```
mysql> select * from bill;
+--------+--------+------------+------------+
| custid | prodid | billamount | billdate   |
+--------+--------+------------+------------+
|      1 |    101 |   16000.85 | 2022-07-28 |
|      4 |    101 |   16000.85 | 2022-07-28 |
|      3 |    103 |   26000.45 | 2022-06-15 |
|      3 |    101 |   15500.00 | 2022-05-22 |
|      3 |    102 |   52000.75 | 2022-07-14 |
+--------+--------+------------+------------+
5 rows in set (0.00 sec)
```

```
mysql> select cust_det.custname,bill.billamount from bill INNER JOIN cust_det on bill
.custid=cust_det.custid ;
+----------+------------+
| custname | billamount |
+----------+------------+
| Abijith  |   16000.85 |
| Raghu    |   16000.85 |
| Murali   |   26000.45 |
| Murali   |   15500.00 |
| Murali   |   52000.75 |
+----------+------------+
5 rows in set (0.00 sec)

mysql> select cust_det.custname,bill.billamount,product_det.prodname from bill INNER
JOIN cust_det on bill.custid=cust_det.custid INNER JOIN product_det on cust_det.prodi
d = product_det.prodid
    -> ;
+----------+------------+-----------+
| custname | billamount | prodname  |
+----------+------------+-----------+
| Abijith  |   16000.85 | samsung   |
| Raghu    |   16000.85 | samsung   |
| Murali   |   26000.45 | samsungAX |
| Murali   |   15500.00 | samsungAX |
| Murali   |   52000.75 | samsungAX |
+----------+------------+-----------+
```

INNER JOIN

“INNER JOIN combines rows from both the tables if they meet a specified condition”.

MySQL Inner Join type returns the records or rows present in both tables If there is at least one match between columns. Or, we can simply say, MySQL Inner Join returns the rows (or records) present in both tables as long as the condition after the ON Keyword is TRUE.

SYNTAX:

SELECT Table1.Column(s), Table2.Column(s) FROM Table1 INNER JOIN Table2 ON Table1.Common_Column = Table2.Common_Column

--OR We can Simply Write it as

SELECT Table1. Column(s), Table2. Column(s) FROM Table1 JOIN Table2 ON Table1.Common_Column = Table2.Common_Column

INNER JOIN

```
mysql> select product_det.prodid,product_det.prodname,cust_det.custname from product_
det INNER JOIN cust_det on product_det.prodid = cust_det.prodid;
+--------+-----------+----------+
| prodid | prodname  | custname |
+--------+-----------+----------+
|    101 | samsung   | Abijith  |
|    103 | samsungAX | Murali   |
|    101 | samsung   | Raghu    |
|    104 | DELL      | Rekha    |
|    105 | ASUS      | Hari     |
|    103 | samsungAX | Yasin    |
+--------+-----------+----------+
6 rows in set (0.00 sec)
```

INNER JOIN

```
mysql> select p.prodid,c.prodid,p.prodname,c.custid,c.custname,p.price from product_d
et AS p INNER JOIN cust_det AS c on p.prodid = c.prodid;
+--------+--------+-----------+--------+----------+----------+
| prodid | prodid | prodname  | custid | custname | price    |
+--------+--------+-----------+--------+----------+----------+
|    101 |    101 | samsung   |      1 | Abijith  | 15000.25 |
|    101 |    101 | samsung   |      4 | Raghu    | 15000.25 |
|    103 |    103 | samsungAX |      3 | Murali   | 25000.75 |
|    103 |    103 | samsungAX |   1235 | Yasin    | 25000.75 |
|    104 |    104 | DELL      |      5 | Rekha    | 35000.25 |
|    105 |    105 | ASUS      |     10 | Hari     | 28000.63 |
+--------+--------+-----------+--------+----------+----------+
6 rows in set (0.00 sec)
```

MySQL Right Join

- MySQL Right Outer Join is one of the Types, which is useful to return all the existing records (or rows) from the Right table, and match rows from the left table. All the Unmatched rows from the left table are filled with NULL Values.
- The MySQL Right Outer join also called Right Join. So it is optional for you to use the Outer Keyword

SYNTAX:

SELECT Table1.Column(s), Table2.Column(s) FROM Table1 RIGHT JOIN Table2 ON Table1.Common_Column = Table2.Common_Column

--OR We can Simply Write it as

SELECT Table1. Column(s), Table2. Column(s) FROM Table1 RIGHT OUTER JOIN Table2 ON Table1.Common_Column = Table2.Common_Column

MySQL Right Join

```
mysql> select * from product_det;
+--------+-----------+----------+
| prodid | prodname  | price    |
+--------+-----------+----------+
|    101 | samsung   | 15000.25 |
|    102 | HP        | 50000.25 |
|    103 | samsungAX | 25000.75 |
|    104 | DELL      | 35000.25 |
|    105 | ASUS      | 28000.63 |
|    106 | samsungDX | 36000.90 |
+--------+-----------+----------+
6 rows in set (0.00 sec)

mysql> select * from cust_det;
+--------+----------+--------+
| custid | custname | prodid |
+--------+----------+--------+
|      1 | Abijith  |    101 |
|      3 | Murali   |    103 |
|      4 | Raghu    |    101 |
|      5 | Rekha    |    104 |
|     10 | Hari     |    105 |
|   1235 | Yasin    |    103 |
+--------+----------+--------+
6 rows in set (0.00 sec)
```

MySQL Right Join

```
mysql> select p.prodid,p.prodname,c.prodid,c.custid,c.custname,p.price from product_d
et AS p RIGHT JOIN cust_det AS c on p.prodid = c.prodid;
+--------+-----------+--------+--------+----------+----------+
| prodid | prodname  | prodid | custid | custname | price    |
+--------+-----------+--------+--------+----------+----------+
|    101 | samsung   |    101 |      1 | Abijith  | 15000.25 |
|    103 | samsungAX |    103 |      3 | Murali   | 25000.75 |
|    101 | samsung   |    101 |      4 | Raghu    | 15000.25 |
|    104 | DELL      |    104 |      5 | Rekha    | 35000.25 |
|    105 | ASUS      |    105 |     10 | Hari     | 28000.63 |
|    103 | samsungAX |    103 |   1235 | Yasin    | 25000.75 |
+--------+-----------+--------+--------+----------+----------+
6 rows in set (0.00 sec)
```

MySQL LEFT Join

- The MySQL Left outer join is to return all the records (or rows) from the first table, and match rows from the right table.

SYNTAX:

SELECT Table1.Column(s), Table2.Column(s) FROM Table1 LEFT JOIN Table2 ON Table1.Common_Column = Table2.Common_Column

--OR We can Simply Write it as

SELECT Table1. Column(s), Table2. Column(s) FROM Table1 LEFT OUTER JOIN Table2 ON Table1.Common_Column = Table2.Common_Column

MySQL LEFT Join

```
mysql> select p.prodid,p.prodname,c.custid,c.custname from product_det AS p lEFT JOIN
 cust_det AS c on p.prodid = c.prodid;
+--------+----------+--------+----------+
| prodid | prodname | custid | custname |
+--------+----------+--------+----------+
|    101 | samsung  |      1 | Abijith  |
|    101 | samsung  |      4 | Raghu    |
|    102 | HP       |   NULL | NULL     |
|    103 | samsungAX |      3 | Murali   |
|    103 | samsungAX |   1235 | Yasin    |
|    104 | DELL     |      5 | Rekha    |
|    105 | ASUS     |     10 | Hari     |
|    106 | samsungDX |   NULL | NULL     |
+--------+----------+--------+----------+
8 rows in set (0.00 sec)
```

CROSS Join

- MySQL Cross Join returns the Cartesian product of both the tables. The Cross Join in MySQL does not require any common column to join two tables.
- The Cartesian product means Number of Rows present in Table 1 Multiplied by Number of Rows present in Table 2.

SYNTAX:

SELECT Table1.Column(s), Table2.Column(s),
FROM Table1 CROSS JOIN Table2

--OR We can Simply Write it as

SELECT Table1. Column(s), Table2. Column(s),
FROM Table1, Table2

CROSS Join

```
mysql> select c.custid,c.custname,b.prodid,b.billamount,b.billdate from bill as b CROSS JOIN cust_det as c on b.custid =
 c.custid;
+--------+----------+--------+-----------+-----------+
| custid | custname | prodid | billamount | billdate  |
+--------+----------+--------+-----------+-----------+
|      1 | Abijith  |    101 |   16000.85 | 2022-07-28 |
|      4 | Raghu    |    101 |   16000.85 | 2022-07-28 |
|      3 | Murali   |    103 |   26000.45 | 2022-07-15 |
|      3 | Murali   |    101 |   15500.00 | 2022-05-22 |
|      3 | Murali   |    102 |   52000.75 | 2022-07-14 |
+--------+----------+--------+-----------+-----------+
5 rows in set (0.00 sec)
```

INNER JOIN – Where Clause

Which customer purchased more products?
Join from Product_det, Cust_det and Bill tables

```
mysql> select cust_det.custname,bill.billamount,product_det.prodname from bill INNER
JOIN cust_det on bill.custid=cust_det.custid INNER JOIN product_det on cust_det.prodi
d = product_det.prodid where cust_det.custid = 3;
+----------+------------+-----------+
| custname | billamount | prodname  |
+----------+------------+-----------+
| Murali   |   26000.45 | samsungAX |
| Murali   |   15500.00 | samsungAX |
| Murali   |   52000.75 | samsungAX |
+----------+------------+-----------+
3 rows in set (0.00 sec)
```

INNER JOIN – Where Clause

Which customer purchased more products ? – Retrieval of data between their cust id 1 and 3; Join from Product_det, Cust_det and Bill tables

```
mysql> select cust_det.custname,bill.billamount,product_det.prodname from bill INNER
JOIN cust_det on bill.custid=cust_det.custid INNER JOIN product_det on cust_det.prodi
d = product_det.prodid where cust_det.custid between 1 and 3 order by cust_det.custna
me;
+----------+------------+-----------+
| custname | billamount | prodname  |
+----------+------------+-----------+
| Abijith  |   16000.85 | samsung   |
| Murali   |   26000.45 | samsungAX |
| Murali   |   15500.00 | samsungAX |
| Murali   |   52000.75 | samsungAX |
+----------+------------+-----------+
4 rows in set (0.04 sec)
```

LEFT JOIN –AGGREGATION – MAX()

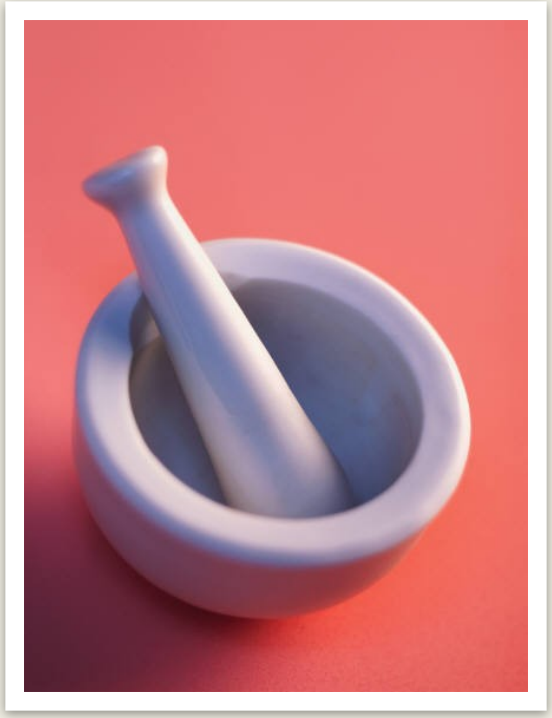
Which customer PURCHASED HIGHEST PRICE PRODUCT? – Retrieval of data FROM two tables ;

```
mysql> select c.custname AS CUSTOMER_NAME, max(b.billamount) as MAXIMUM_PRICE from cu
st_det as c LEFT JOIN bill as b on c.custid = b.custid;
+---------------+---------------+
| CUSTOMER_NAME | MAXIMUM_PRICE |
+---------------+---------------+
| Abijith       |      52000.75 |
+---------------+---------------+
1 row in set (0.03 sec)
```

LEFT JOIN –AGGREGATION.

CUSTOMER DETAILS : WHO BOUGHT PRODUCT AND NOT?

```
mysql> select c.custname as CUSTOMER_NAME,b.billamount as BILL_AMOUNT from cust_det a
s c LEFT JOIN bill as b on c.custid = b.custid;
+---------------+-------------+
| CUSTOMER_NAME | BILL_AMOUNT |
+---------------+-------------+
| Abijith       |    16000.85 |
| Murali        |    26000.45 |
| Murali        |    15500.00 |
| Murali        |    52000.75 |
| Raghu         |    16000.85 |
| Rekha         |        NULL |
| Hari          |        NULL |
| Yasin         |        NULL |
+---------------+-------------+
8 rows in set (0.00 sec)
```



Thank You