CHOCOLATE_FUSION DATABASE PROJECT

Overview of the Project:

The **Chocolate_Fusion** database project is designed to facilitate the management of key business information for a chocolate company. In today's competitive market, efficient data management is crucial for operational success, customer satisfaction, and informed decision-making. This database provides a structured approach to store and retrieve essential data related to various aspects of the business.

Purpose:

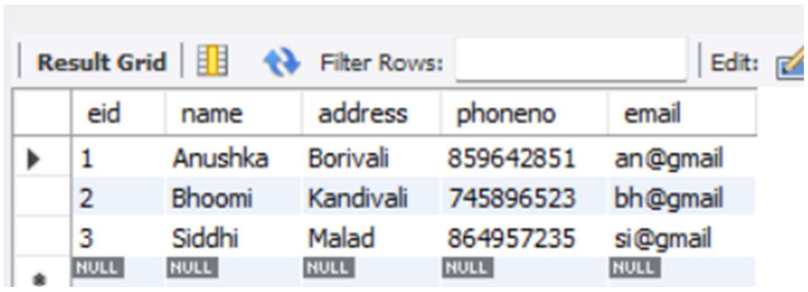
To manage information for a chocolate business including staff, products, customers, and sales orders.

Database Structure:

```
create database Chocolate_Fusion;

use Chocolate_Fusion;


create table staff
(eid int auto_increment primary key     ,
name varchar (20),
address varchar (20) not null,
phoneno double,
email varchar (30) not null unique);

create table branch
(id int,
Bname varchar (20));

create table product
(Pid int primary key,
Pname varchar (20));

create table customer
(Cid int auto_increment primary key,
name varchar (20),
location varchar (20),
phoneno double );


create table sales_order (
id int,
amount double,
boxes int,
Cid int,
Pid int,
foreign key (Cid) references customer (Cid),
foreign key (Pid) references Product (Pid));


insert into staff
(name, address, phoneno ,email)
values ("Anushka", "Borivali", 859642851,"an@gmail"),
("Bhoomi", "Kandivali", 745896523,"bh@gmail"),
("Siddhi", "Malad", 864957235, "si@gmail");

insert into branch
(id, Bname)
values (1,"dadar"),
(2,"thane"),
(3,"virar");

insert into product
(Pid, Pname)
values (01, "Whitechoc"),
(02, "milk bar"),
(03, "dark & pure"),
(04, "caramel stuffed bars");

insert into customer
(name, location, phoneno)
values ("Reena","mumbai",8564259685),
("Meena","mumbai",8695324517),
("Teena","thane",9586324516),
("Rani","virar",8641257395),
("Pinku","virar",7598632145 );

insert into sales_order
(id, amount, boxes, Cid, Pid)
Values (1,4000,5,5,01),
(2,2000,3,5,03),
(3,7500,6,1,03),
(4,9500,8,3,02),
(5,6500,5,4,02),
(6,1200,2,1,01),
(7,4500,3,2,04),
(8,3000,4,4,04),
(9,4100,3,1,01);
```

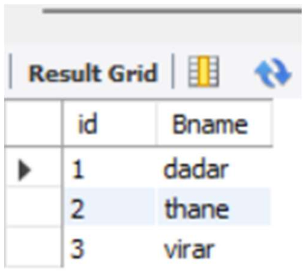
- **This query is showing all the tables**

```
select * from staff;
```



eid	name	address	phoneno	email
1	Anushka	Borivali	859642851	an@gmail
2	Bhoomi	Kandivali	745896523	bh@gmail
3	Siddhi	Malad	864957235	si@gmail
NULL	NULL	NULL	NULL	NULL

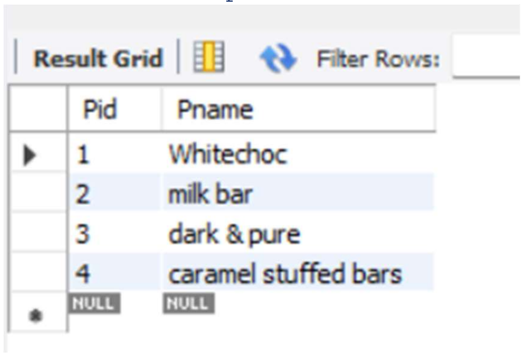
```
select * from branch;
```



Result Grid

id	Bname
1	dadar
2	thane
3	virar

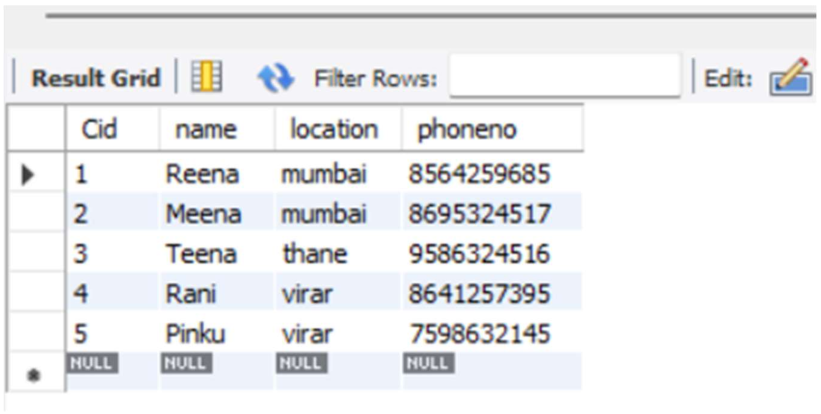
```
select * from product;
```



Result Grid | Filter Rows:

Pid	Pname
1	Whitechoc
2	milk bar
3	dark & pure
4	caramel stuffed bars
NULL	NULL

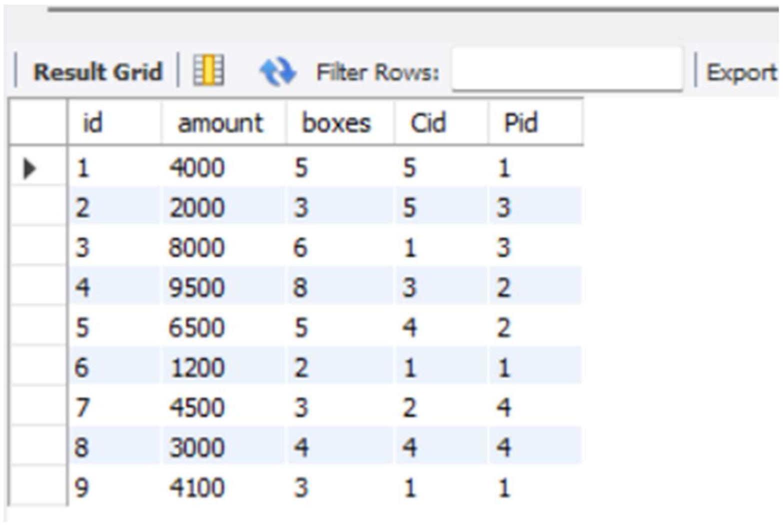
```
select * from customer;
```



Result Grid | Filter Rows: | Edit:

Cid	name	location	phoneno
1	Reena	mumbai	8564259685
2	Meena	mumbai	8695324517
3	Teena	thane	9586324516
4	Rani	virar	8641257395
5	Pinku	virar	7598632145
NULL	NULL	NULL	NULL

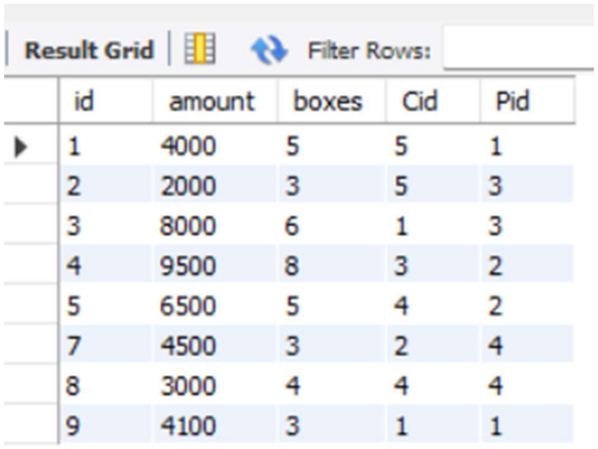
```
select * from sales_order;
```



id	amount	boxes	Cid	Pid
1	4000	5	5	1
2	2000	3	5	3
3	8000	6	1	3
4	9500	8	3	2
5	6500	5	4	2
6	1200	2	1	1
7	4500	3	2	4
8	3000	4	4	4
9	4100	3	1	1

- **In the query we can see that the orders amount are less than 1200**

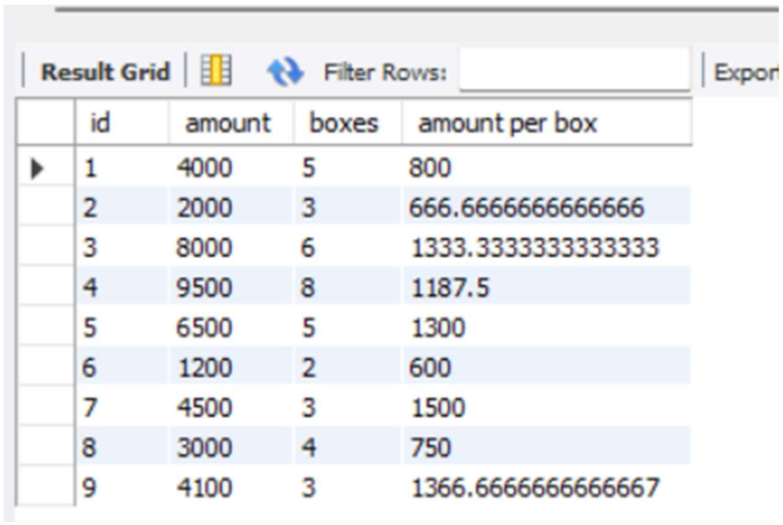
```
select * from sales_order where amount > 1200;
```



id	amount	boxes	Cid	Pid
1	4000	5	5	1
2	2000	3	5	3
3	8000	6	1	3
4	9500	8	3	2
5	6500	5	4	2
7	4500	3	2	4
8	3000	4	4	4
9	4100	3	1	1

- **The below query provides a summary of sales orders, highlighting how much revenue is generated per box for each order.**

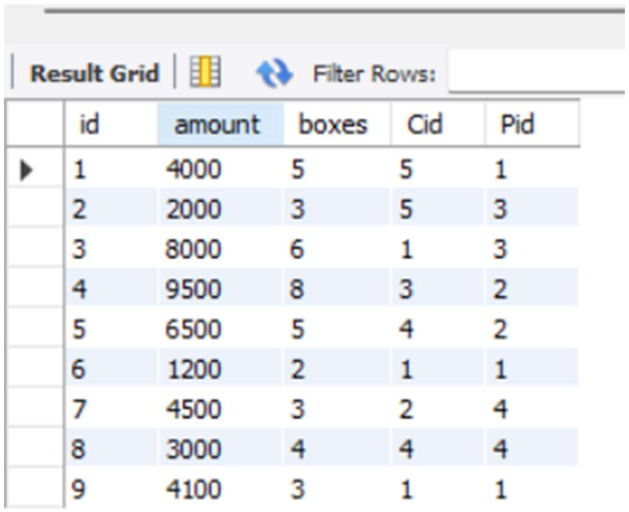
```
select id, amount, boxes, amount/boxes as 'amount per box' from
sales_order;
```


Result Grid | Filter Rows: | Export

	id	amount	boxes	amount per box
▶	1	4000	5	800
	2	2000	3	666.6666666666666
	3	8000	6	1333.3333333333333
	4	9500	8	1187.5
	5	6500	5	1300
	6	1200	2	600
	7	4500	3	1500
	8	3000	4	750
	9	4100	3	1366.6666666666667

- **The below combined query allows you to update a specific sales order and then view the entire table to confirm the update.**

```
update sales_order
set amount = 8000
where id = 3;
select * from sales_order;
```


Result Grid | Filter Rows:

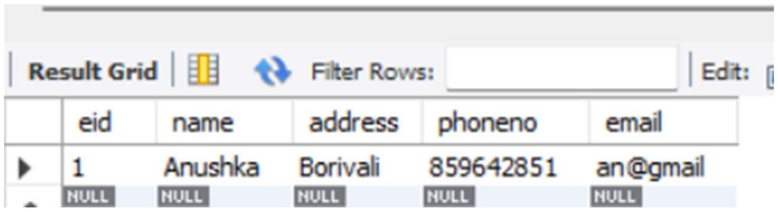
	id	amount	boxes	Cid	Pid
▶	1	4000	5	5	1
	2	2000	3	5	3
	3	8000	6	1	3
	4	9500	8	3	2
	5	6500	5	4	2
	6	1200	2	1	1
	7	4500	3	2	4
	8	3000	4	4	4
	9	4100	3	1	1

- **disables safe updates in MySQL, allowing you to run UPDATE and DELETE commands without requiring a WHERE clause to limit affected rows.**

```
set sql_safe_updates = 0;
```

- **The retrieves all columns and rows from the staff table where the name starts with the letter 'a'. The % wildcard matches any sequence of characters following 'a'.**

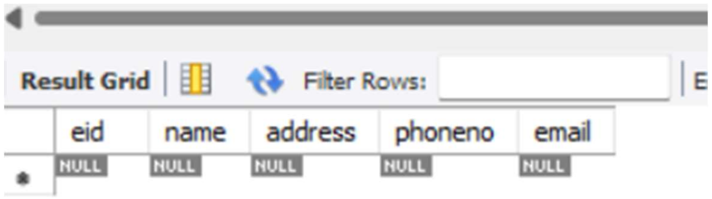
```
select * from staff where name like 'a%';
```



eid	name	address	phoneno	email
1	Anushka	Borivali	859642851	an@gmail
NULL	NULL	NULL	NULL	NULL

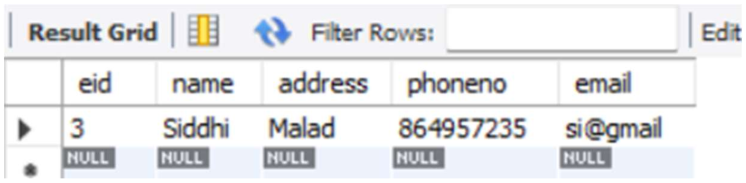
- **In SQL, the AND and OR operators are used to combine multiple conditions in a WHERE clause, allowing for more complex filtering of records. Here's how they work:**

```
select * from staff where eid = 3 and "location" = "virar";
```



eid	name	address	phoneno	email
NULL	NULL	NULL	NULL	NULL

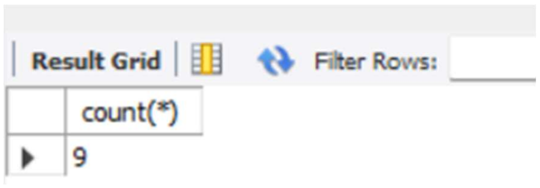
```
select * from staff where eid = 3 or "location" = "virar";
```



eid	name	address	phoneno	email
3	Siddhi	Malad	864957235	si@gmail
NULL	NULL	NULL	NULL	NULL

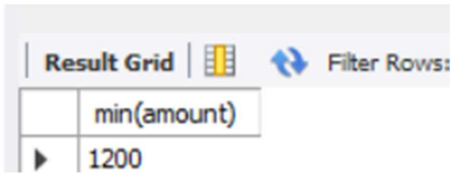
- **Common Aggregate Functions**

```
select count(*) from sales_order;
```



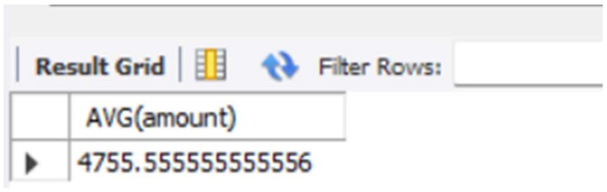
count(*)
9

```
select min(amount) from sales_order;
```



min(amount)
1200

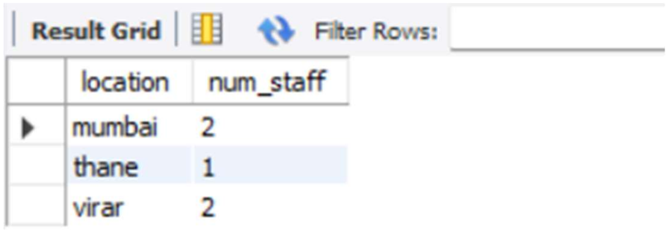
```
SELECT AVG(amount) FROM sales_order;
```



	AVG(amount)
▶	4755.555555555556

- **Aggregate functions are often used with the GROUP BY clause to group rows that have the same values in specified columns:**

```
SELECT location, COUNT (*) AS num_staff
FROM customer
GROUP BY location;
```



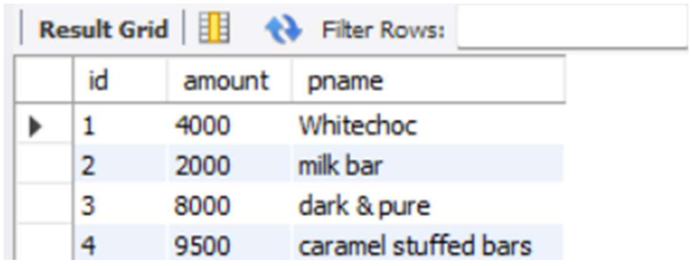
	location	num_staff
▶	mumbai	2
	thane	1
	virar	2

- **JOINS**

Inner join

This query retrieves the id and amount from the sales_order table and the pname from the product table by performing an inner join where the product ID (Pid) matches the sales order ID (id).

```
select s.id, s.amount, p.pname from sales_order s join product p on p.Pid
= s.id;
```

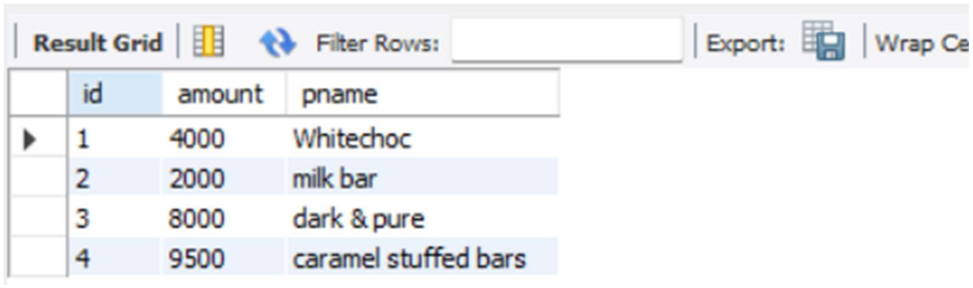


	id	amount	pname
▶	1	4000	Whitechoc
	2	2000	milk bar
	3	8000	dark & pure
	4	9500	caramel stuffed bars

Left Join

This query retrieves the id and amount from the sales_order table and the pname from the product table, including all products even if there are no matching sales orders, with NULL values for sales order columns where there is no match.

```
select s.id, s.amount, p.pname from product p left join sales_order s on
s.id = p.pid;
```

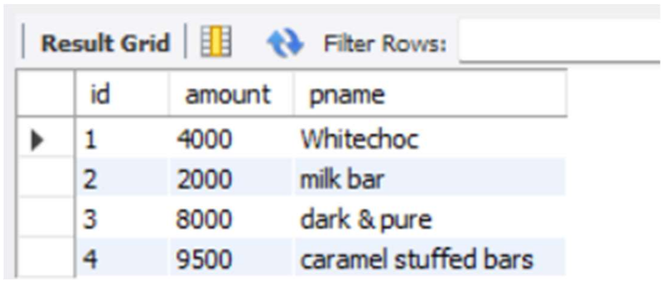


id	amount	pname
1	4000	Whitechoc
2	2000	milk bar
3	8000	dark & pure
4	9500	caramel stuffed bars

Right Join

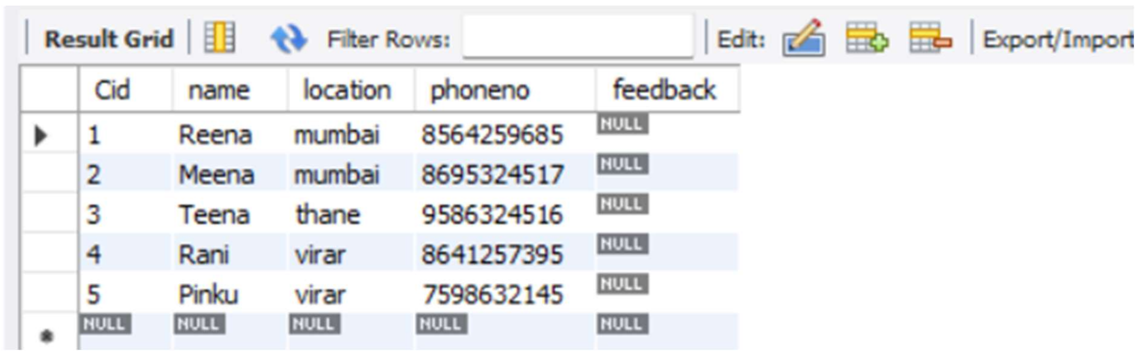
This query performs a right join between the sales_order and product tables, returning all products even if they have no corresponding sales orders. It selects the product ID, amount from sales orders, and product name.

```
select s.id, s.amount, p.pname from sales_order s right join product p on
p.pid = s.id;
```



id	amount	pname
1	4000	Whitechoc
2	2000	milk bar
3	8000	dark & pure
4	9500	caramel stuffed bars

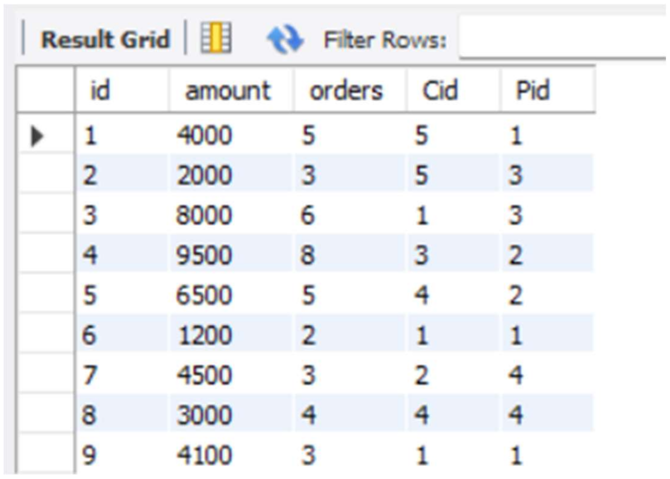
```
alter table customer
add column feedback varchar (20);

select * FROM customer;
```



Cid	name	location	phoneno	feedback
1	Reena	mumbai	8564259685	NULL
2	Meena	mumbai	8695324517	NULL
3	Teena	thane	9586324516	NULL
4	Rani	virar	8641257395	NULL
5	Pinku	virar	7598632145	NULL
NULL	NULL	NULL	NULL	NULL

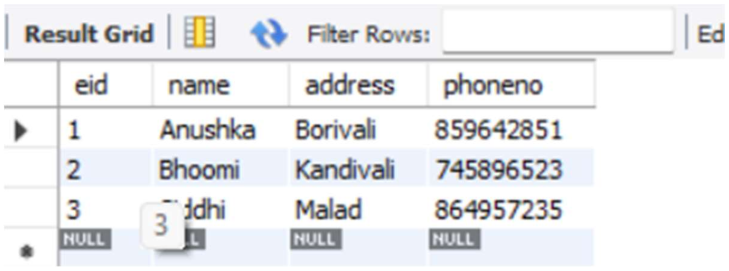
```
alter table sales_order
rename column boxes to orders;
```

```
select * FROM sales_order;
```



id	amount	orders	Cid	Pid
1	4000	5	5	1
2	2000	3	5	3
3	8000	6	1	3
4	9500	8	3	2
5	6500	5	4	2
6	1200	2	1	1
7	4500	3	2	4
8	3000	4	4	4
9	4100	3	1	1

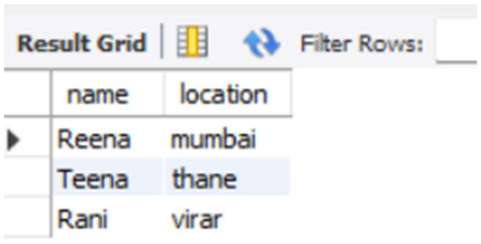
```
alter table staff
drop column email;

select * FROM staff;
```



eid	name	address	phoneno
1	Anushka	Borivali	859642851
2	Bhoomi	Kandivali	745896523
3	Siddhi	Malad	864957235
NULL	NULL	NULL	NULL

This subquery finds customers who have placed orders with an amount greater than the average order amount.

```
SELECT DISTINCT c.name, c.location
FROM customer c
WHERE c.Cid IN (SELECT so.Cid FROM sales_order so WHERE so.amount >
(SELECT AVG(amount) FROM sales_order));
```

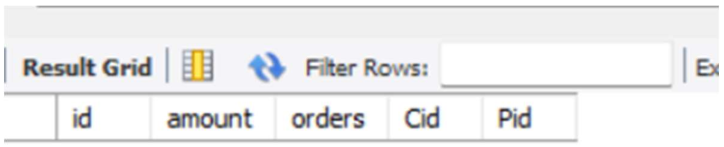


name	location
Reena	mumbai
Teena	thane
Rani	virar

Truncate Queries

TRUNCATE is used to remove all rows from a table without logging individual row deletions, which is faster than DELETE. However, it resets any auto-increment counters.

```
TRUNCATE TABLE sales_order;
select * from sales_order;
```

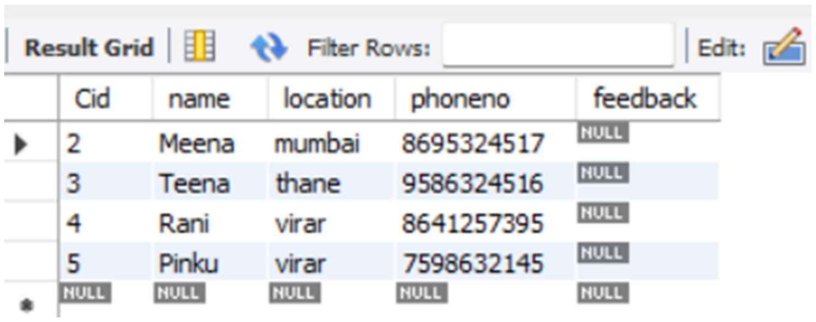


id	amount	orders	Cid	Pid

DELETE Queries

This command can be used to delete specific rows using conditions. For example, to delete a specific customer

```
DELETE FROM customer WHERE Cid = 1;

select * from CUSTOMER;
```



Cid	name	location	phoneno	feedback
2	Meena	mumbai	8695324517	NULL
3	Teena	thane	9586324516	NULL
4	Rani	virar	8641257395	NULL
5	Pinku	virar	7598632145	NULL
NULL	NULL	NULL	NULL	NULL

************************THE END*************************************