CSE202 - Fundamentals of Database Management Systems
Final Project

Online Retail Store

Aviral Dawar - 2020040
Soumya Aggarwal - 2020
Utkarsh Arora - 2020143
Vansh Gandhi - 2020147

Based on the feedback of our midsem evaluation we have included the following in our project:

1. Cascading
2. More complex queries
3. Increased Scope of project

INDEX

Query Optimisation has been done while writing Queries → Ex1 Ex2

Scope of Project

1. We have implemented a retail store in which a customer can buy from many items.
2. Customer can either login from a pre-existing account or create a new one.
3. Products can be of various categories.
4. Items are supplied by supplier who has grant to view all the supplies.
5. Customers can order products which are then delivered by delivery partners
6. Customers can give any rating to products.
7. Customers can raise complaints which can be viewed by customer care.
8. Customers can put products in their cart.
9. Customers can use coupons to get discounts.
10. Customers get details about their bill.

Relational Schema

Account(Account_id,email,password)

Customer(Customer_id, Account_id,First Name,Last Name,House Number,street,city,State,Country,Age,PhoneNumber,cart_cost)

Product(Product_ID, Name, price, Stock, Category_ID)

Rating(Product_ID,Customer_id, rating)

Complaints(Product_ID,Customer_id, complaintText)

Category(Category_ID,Name)

Supplier(Supplier_ID, Account_id,First Name, Last Name,Age, PhoneNumber, House Number,street,city,State,Country)

Supplies(Supplier_ID,Product_ID)

Orders(Customer_ID,DeliveryPartner_ID,Product_ID,Coupon_ID, Status,Quantity,Order Date)

Delivery Partner(DeliveryPartner_ID, Account_id, First Name,Last Name,House Number,street,city,State, Country,PhoneNumber)

Cart(Product_id,customer_id, quantity)

Customer_Coupon(coupon_id, customer_id)

Coupon(coupon_id, coupon_text,valid_upto,pct_disc)

Views

View 1 : For Customers to see their Account details in Profile

```
create view Customer_Details as
select a.email as email_address, c.first_name as first_name, c.last_name as
last_name, c.age as age, c.cart_cost as cart_cost
from Account as a, Customer as c
where a.account_id = c.account_id
```

email_address	first_name	last_name	age	cart_cost
aibkmklh@gmail.com	Evangelina	Landis	74	0.00
aidl@gmail.com	Mark	Armstrong	16	0.00
andjujuhqex@gmail.com	James	Kim	37	0.00
anxczkc@gmail.com	Christine	Lehman	16	0.00
aqxizut@gmail.com	Aaron	Jacks	24	0.00
asshakbijtlb@gmail.com	Billy	Yedinak	30	0.00
avaikzlg@gmail.com	Annie	Gates	17	0.00
azzuy@gmail.com	Randy	Brown	54	0.00
bgmhzyinwecl@gmail.com	Rita	Workman	68	0.00
biawvrrydob@gmail.com	Janita	Utterback	68	0.00
blsi@gmail.com	Bonnie	Reynolds	85	0.00
bsaxienq@gmail.com	Diane	Moore	45	0.00
bstuicvjxfx@gmail.com	Robert	Moldenha...	63	0.00
bvvuw@gmail.com	Carmine	Dorsey	17	0.00
bwnzraxn@gmail.com	Chrissy	Taft	19	0.00
cbxhkcbwl@gmail.com	Lorene	Sullivan	19	0.00
ckcz@gmail.com	Robert	Rison	24	0.00
clcfmjx@gmail.com	Margie	Thielman	74	0.00
cmdoamaafni@gmail.com	Audrey	Serviss	52	0.00
coukrqqjix@gmail.com	Chris	Tison	51	0.00
cvww@gmail.com	Mellie	Dawson	77	0.00
cyjwq@gmail.com	Clarice	Kirsch	50	0.00
cyugrxodngps@gmail.com	Mel	Ashley	53	0.00

View 2: For Suppliers/Customers to see all products

```
create view product_details as
select product.name as Product_Name, product.price as Price, category.name as
Product_Category
from product
inner join category
on product.category_id = category.category_id
```

Product_Name	Price	Product_Category
Apple	55.00	Fruit
Banana	29.00	Fruit
Pear	25.00	Fruit
Strawberry	49.00	Fruit
Watermelon	46.00	Fruit
Muskmelon	65.00	Fruit
Grape	52.00	Fruit
Kiwi	84.00	Fruit
Potato	41.00	Vegetable
Cucumber	28.00	Vegetable
Kale	86.00	Vegetable
Lettuce	39.00	Vegetable
Apple Juice	39.00	Juice
Mango Juice	46.00	Juice
Strawberry J...	46.00	Juice
Mixed Fruit Ju...	44.00	Juice
Milk	90.00	Dairy
Cheese	61.00	Dairy
Curd	86.00	Dairy
Greek Yogurt	72.00	Dairy
Yogurt	36.00	Dairy
Paneer	74.00	Dairy
Oreo	21.00	Biscuit

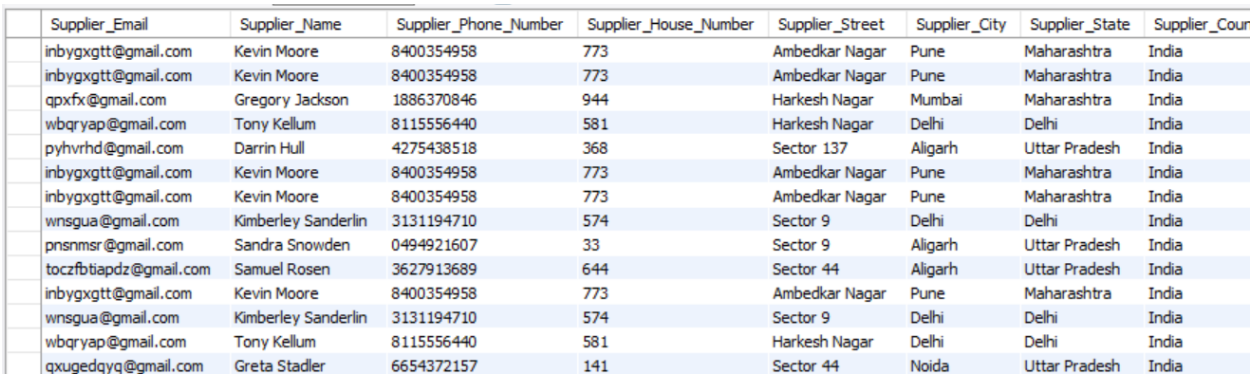
View 3: For Customer Care to see complaints and customer with PhoneNo.

```
create view ProductComplaints as
select c.first_name as CustName , c.phone_number as PhoneNumber,
com.complaint_text as Complaint,p.name as ProductName
from Customer as c , Complaints as com , Product as p
where c.customer_id = com.customer_id and com.product_id = p.product_id
```

CustName	PhoneNumber	Complaint	ProductName
Dana	9781326550	supplier sent hitmen to my house	Kiwi
Margie	0890348819	Bad Colour	Pear
Jeanna	1505570237	Me do not likey	Lettuce
Janita	4597929190	Me do not likey	Milk
Dorothy	1008220503	supplier sent hitmen to my house	Potato
Albert	4845322252	Bad	Yogurt
James	1806947222	Bad Picture	Lettuce
Randy	6677837217	Bad Colour	Milk
Emma	3638503303	supplier sent hitmen to my house	Greek Yogurt
Maxine	8522124846	Very Bad	Muskmelon
Jorge	9141718220	Bad Colour	Apple
Viola	4323683399	Too small	Parle-G
Eric	2061081269	supplier sent hitmen to my house	Kiwi
Annette	1516607731	Me do not likey	Greek Yogurt
Kenneth	8803111736	Very Bad	Watermelon
Felisa	0972111999	Very Bad	Pork
Ronald	5628980455	Bad	Bourboun
Bessie	8237963584	Bad Colour	Bourboun
Megan	0164192372	Broken	Paneer
Sandy	4300127705	supplier sent hitmen to my house	Pork
Amanda	3913676942	Bad Colour	Yogurt
Dorothy	1008220503	supplier sent hitmen to my house	Paneer
Emma	3638503303	Bad Picture	Mango Juice

View 4: For Delivery Partner to see details of order they are delivering

```
create view order_details as
select a1.email as Supplier_Email, CONCAT(s.first_name,' ',s.last_name) as
Supplier_Name,
s.phone_number as Supplier_Phone_Number, s.house_number as
Supplier_House_Number,
s.street as Supplier_Street, s.city as Supplier_City, s.state as Supplier_State,
s.country as Supplier_Country,
p.name as Product_Name, o.quantity as Product_Quantity,
(o.quantity*p.price*(100-cp.pct_discount)/100) as Final_Price,
a2.email as Customer_Email, CONCAT(c.first_name,' ',c.last_name) as
Customer_Name,
c.phone_number as Customer_Phone_Number, c.house_number as
Customer_House_Number, c.street as Customer_Street,
c.city as Customer_City, c.state as Customer_State, c.country as Customer_Country
from Account as a1, Supplier as s, Product as p, Orders as o, Coupon as cp, Account
as a2, Customer as c, supplies
where a1.account_id = s.account_id
and s.supplier_id = supplies.supplier_id
and supplies.product_id = p.product_id
and p.product_id = o.product_id
and cp.coupon_id = o.coupon_id
and c.customer_id = o.customer_id
and a2.account_id = c.account_id
and o.delivery_partner_id = {}
```



Supplier_Email	Supplier_Name	Supplier_Phone_Number	Supplier_House_Number	Supplier_Street	Supplier_City	Supplier_State	Supplier_Cour
inbygxgtt@gmail.com	Kevin Moore	8400354958	773	Ambedkar Nagar	Pune	Maharashtra	India
inbygxgtt@gmail.com	Kevin Moore	8400354958	773	Ambedkar Nagar	Pune	Maharashtra	India
qpxfx@gmail.com	Gregory Jackson	1886370846	944	Harkesh Nagar	Mumbai	Maharashtra	India
wbqryap@gmail.com	Tony Kellum	8115556440	581	Harkesh Nagar	Delhi	Delhi	India
pyhvrhd@gmail.com	Darrin Hull	4275438518	368	Sector 137	Aligarh	Uttar Pradesh	India
inbygxgtt@gmail.com	Kevin Moore	8400354958	773	Ambedkar Nagar	Pune	Maharashtra	India
inbygxgtt@gmail.com	Kevin Moore	8400354958	773	Ambedkar Nagar	Pune	Maharashtra	India
wnsgua@gmail.com	Kimberley Sanderlin	3131194710	574	Sector 9	Delhi	Delhi	India
pnsnmsr@gmail.com	Sandra Snowden	0494921607	33	Sector 9	Aligarh	Uttar Pradesh	India
toczfbtiapdz@gmail.com	Samuel Rosen	3627913689	644	Sector 44	Aligarh	Uttar Pradesh	India
inbygxgtt@gmail.com	Kevin Moore	8400354958	773	Ambedkar Nagar	Pune	Maharashtra	India
wnsgua@gmail.com	Kimberley Sanderlin	3131194710	574	Sector 9	Delhi	Delhi	India
wbqryap@gmail.com	Tony Kellum	8115556440	581	Harkesh Nagar	Delhi	Delhi	India
qxugedqyq@gmail.com	Greta Stadler	6654372157	141	Sector 44	Noida	Uttar Pradesh	India

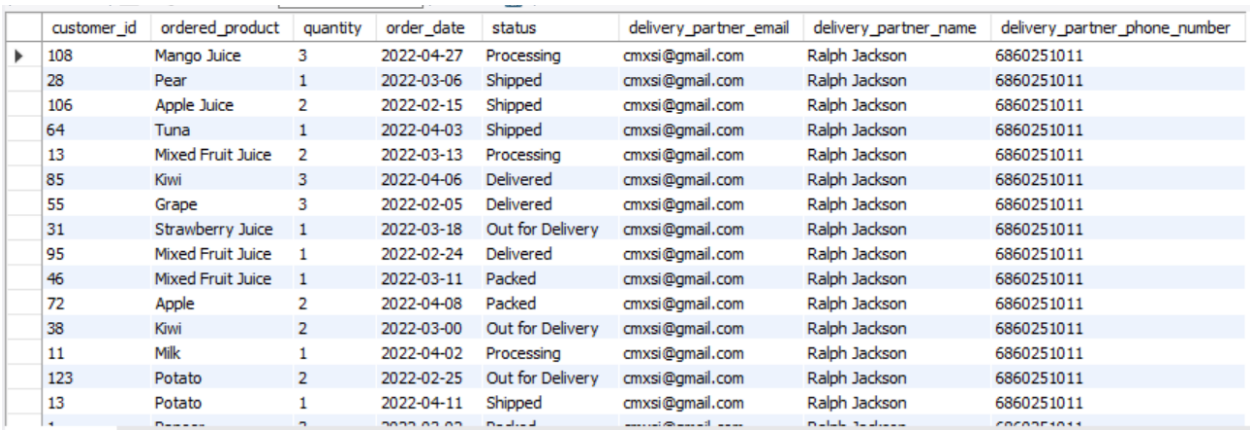
Supplier_Country	Product_Name	Product_Quantity	Final_Price	Customer_Email	Customer_Name	Customer_Phone_Number	Customer_Hou
India	Mixed Fruit Juice	1	14.400000	ecnynpkaexc@gmail.com	Mike Bala	1277611630	856
India	Mixed Fruit Juice	1	14.400000	dhpb@gmail.com	Joan Delvalle	8553203787	304
India	Apple	2	111.600000	cssvaszozjm@gmail.com	Jose Michaelson	2562998157	813
India	Kiwi	2	39.000000	xytgjf@gmail.com	Lorenzo Thomas	3582725266	161
India	Milk	1	27.000000	uzdrkq@gmail.com	Melba Wilson	3279539815	736
India	Potato	2	80.000000	qijwrclibwf@gmail.com	Patricia Hess	2680819862	310
India	Potato	1	25.000000	covzduxoif@gmail.com	John Evans	1384653139	651
India	Paneer	3	202.500000	ntkafktlvac@gmail.com	Helen Peterson	1939653135	737
India	Muskmelon	3	117.000000	scrjxjqho@gmail.com	Diane Hampton	7850938261	729
India	Yogurt	3	109.500000	bnxj@gmail.com	Nina Williams	7745664508	316
India	Mixed Fruit Juice	3	115.200000	patblczoc@gmail.com	Willie Pleasant	7826251717	958
India	Paneer	3	270.000000	qbsbb@gmail.com	David Wilkes	7047157640	212
India	Kiwi	1	39.000000	fslkmk@gmail.com	Raymond Henry	6444393977	283
India	Apple Juice	2	64.000000	enodus@gmail.com	Evelyn Chapman	2478934498	726

Email	Customer_Name	Customer_Phone_Number	Customer_House_Number	Customer_Street	Customer_City	Customer_State	Customer_Country
c@gmail.com	Mike Bala	1277611630	856	Harkesh Nagar	Delhi	Delhi	India
.com	Joan Delvalle	8553203787	304	Harkesh Nagar	Pune	Maharashtra	India
@gmail.com	Jose Michaelson	2562998157	813	Govindpuri	Delhi	Delhi	India
il.com	Lorenzo Thomas	3582725266	161	Sector 137	Aligarh	Uttar Pradesh	India
ail.com	Melba Wilson	3279539815	736	Sector 137	Aligarh	Uttar Pradesh	India
gmail.com	Patricia Hess	2680819862	310	Ambedkar Nagar	Aligarh	Uttar Pradesh	India
gmail.com	John Evans	1384653139	651	Sector 137	Mumbai	Maharashtra	India
gmail.com	Helen Peterson	1939653135	737	Sector 44	Aligarh	Uttar Pradesh	India
mail.com	Diane Hampton	7850938261	729	Harkesh Nagar	Noida	Uttar Pradesh	India
com	Nina Williams	7745664508	316	Ambedkar Nagar	Delhi	Delhi	India
gmail.com	Willie Pleasant	7826251717	958	Harkesh Nagar	Aligarh	Uttar Pradesh	India
il.com	David Wilkes	7047157640	212	Sector 137	Noida	Uttar Pradesh	India
il.com	Raymond Henry	6444393977	7047157640	Ambedkar Nagar	Noida	Uttar Pradesh	India
ail.com	Evelyn Chapman	2478934498	726	Govindpuri	Delhi	Delhi	India

View 5: View to show details that a customer gets to see on their order regarding delivery partner.

```
create view_customer_order_delivery_partner as
select o.customer_id, p.name as ordered_product,
o.quantity as quantity, o.order_date as order_date,
o.status as status, a.email as delivery_partner_email,
CONCAT(dp.first_name,' ',dp.last_name) as
delivery_partner_name,
dp.phone_number as delivery_partner_phone_number
```

```
from product as p, orders as o, account as a,
delivery_partner as dp
where p.product_id = o.product_id
and o.delivery_partner_id = dp.delivery_partner_id
and dp.account_id = a.account_id
```



customer_id	ordered_product	quantity	order_date	status	delivery_partner_email	delivery_partner_name	delivery_partner_phone_number
108	Mango Juice	3	2022-04-27	Processing	cmxsi@gmail.com	Ralph Jackson	6860251011
28	Pear	1	2022-03-06	Shipped	cmxsi@gmail.com	Ralph Jackson	6860251011
106	Apple Juice	2	2022-02-15	Shipped	cmxsi@gmail.com	Ralph Jackson	6860251011
64	Tuna	1	2022-04-03	Shipped	cmxsi@gmail.com	Ralph Jackson	6860251011
13	Mixed Fruit Juice	2	2022-03-13	Processing	cmxsi@gmail.com	Ralph Jackson	6860251011
85	Kiwi	3	2022-04-06	Delivered	cmxsi@gmail.com	Ralph Jackson	6860251011
55	Grape	3	2022-02-05	Delivered	cmxsi@gmail.com	Ralph Jackson	6860251011
31	Strawberry Juice	1	2022-03-18	Out for Delivery	cmxsi@gmail.com	Ralph Jackson	6860251011
95	Mixed Fruit Juice	1	2022-02-24	Delivered	cmxsi@gmail.com	Ralph Jackson	6860251011
46	Mixed Fruit Juice	1	2022-03-11	Packed	cmxsi@gmail.com	Ralph Jackson	6860251011
72	Apple	2	2022-04-08	Packed	cmxsi@gmail.com	Ralph Jackson	6860251011
38	Kiwi	2	2022-03-00	Out for Delivery	cmxsi@gmail.com	Ralph Jackson	6860251011
11	Milk	1	2022-04-02	Processing	cmxsi@gmail.com	Ralph Jackson	6860251011
123	Potato	2	2022-02-25	Out for Delivery	cmxsi@gmail.com	Ralph Jackson	6860251011
13	Potato	1	2022-04-11	Shipped	cmxsi@gmail.com	Ralph Jackson	6860251011

We can later use select command on this using “where customer_id = {}”

View 6: Basic Product Details

```
create view basic_product_details as
select product.name as Product_Name, product.price as
Price, category.name as Product_Category
from product
inner join category
on product.category_id = category.category_id
```

	Product_Name	Price	Product_Category
	Mixed Fruit Ju...	48.00	Juice
	Milk	54.00	Dairy
	Cheese	73.00	Dairy
	Curd	18.00	Dairy
	Greek Yogurt	36.00	Dairy
	Yogurt	73.00	Dairy
	Paneer	90.00	Dairy
	Oreo	21.00	Biscuit
	Hide n Seek	21.00	Biscuit
	Bourboun	24.00	Biscuit
	Parle-G	23.00	Biscuit
	Chicken	114.00	Meat
	Pork	91.00	Meat
	Lamb	72.00	Meat
	Tuna	92.00	Meat

Grants

We have created two users, ‘supplier’ and ‘customer_care’

‘Supplier’ has been granted insert access on ‘supplies’ table.

‘Customer_care’ has been granted insert access on ‘Complaints’ and ‘Ratings’ tables.

Meanwhile, ‘Supplier’ has been granted view access on ‘Complaints’ and ‘Ratings’ tables, while insert access on these tables has been revoked.

1) Granting access to Supplier

```
CREATE USER IF NOT EXISTS 'supplier'@'localhost'
IDENTIFIED BY 'supplier';

GRANT INSERT
ON Supplies
TO 'supplier'@'localhost';

SHOW GRANTS FOR 'supplier'@'localhost'

REVOKE INSERT
ON Supplies
FROM 'supplier'@'localhost';

SHOW GRANTS FOR 'supplier'@'localhost'

DROP USER IF EXISTS 'supplier'@'localhost'
```

2) Granting access to Customer Care

```
CREATE USER IF NOT EXISTS 'customer_care'@'localhost'
IDENTIFIED BY 'customer_care';

GRANT SELECT
ON Complaints
TO 'customer_care'@'localhost';

SHOW GRANTS FOR 'customer_care'@'localhost'

DROP USER IF EXISTS 'customer_care'@'localhost'
```

Queries

Query 1: Lists the income of each supplier through the orders that have been placed for the products they are supplying

```
select s.supplier_id, a.email, concat(s.first_name,' ',s.last_name) as supplier_name,
sum(p.price*o.quantity*c.pct_discount/100) as Income
from supplier as s, account as a, product as p, orders as o, supplies, coupon as c
where s.account_id = a.account_id
and s.supplier_id = supplies.supplier_id
and supplies.product_id = p.product_id
and p.product_id = o.product_id
and o.coupon_id = c.coupon_id
group by s.supplier_id
order by Income desc
```

supplier_id	email	supplier_name	Income
9	wnsgua@gmail.com	Kimberley Sanderlin	184.050000
3	qxugedqyq@gmail.com	Greta Stadler	97.600000
5	toczfbtiapdz@gmail.com	Samuel Rosen	54.000000
7	wbqryap@gmail.com	Tony Kellum	53.700000
10	shnhnabvea@gmail.com	Sibyl Horvath	36.800000
8	inbygxgtt@gmail.com	Kevin Moore	28.400000
1	qpxfx@gmail.com	Gregory Jackson	23.250000

Relational Algebraic Expression

$\tau_{\text{income}\downarrow}$

$\pi_{\text{s . supplier\_id, a . email,s.first\_name+s.last\_name} \rightarrow \text{supplier\_name, SUM (p.price*o.quantity*c.pct\_discount/100)} \rightarrow \text{income}}$

$\gamma_{\text{s.supplier\_id,}}$

$\sigma_{\text{s . account\_id = a . account\_id AND s . supplier\_id = supplies . supplier\_id AND supplies . product\_id = p . product\_id AND p . product\_id = o . product\_id AND o . coupon\_id = c . coupon\_id}}$

$(\rho_s \text{supplier} \times$
$\rho_a \text{account} \times$
$\rho_p \text{product} \times$
$\rho_o \text{orders} \times \text{supplies} \times$
$\rho_c \text{coupon})$

Query 2: Customer with an undelivered order for the longest time.

```
select customer_id, product_id, delivery_partner_id, coupon_id, quantity,
min(order_date)
from orders
where status <> "Delivered"
```

customer_id	product_id	delivery_partner_id	coupon_id	quantity	min(order_date)
91	20	5	6	3	2022-02-00

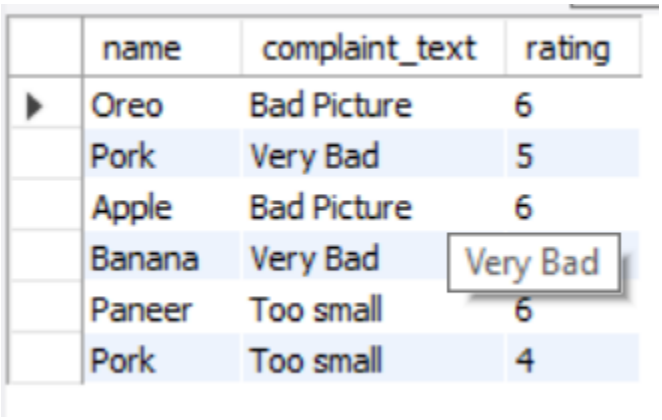
Relational Algebraic Expression

$\pi_{\text{customer\_id, product\_id, delivery\_partner\_id, coupon\_id, quantity, MIN (order\_date)}}$
$\gamma_{\text{MIN (order\_date)}}$
$\sigma_{\text{status <> "Delivered"}} \text{orders}$

Query 3: Lists the complaint text of products that have an accompanying rating to go along with it

```
select product.name, complaints.complaint_text, rating.rating
from product
right join(
complaints
join rating
on complaints.product_id = rating.product_id and complaints.customer_id =
rating.customer_id
```

)
on product.product_id = complaints.product_id



	name	complaint_text	rating
▶	Oreo	Bad Picture	6
	Pork	Very Bad	5
	Apple	Bad Picture	6
	Banana	Very Bad	
	Paneer	Too small	6
	Pork	Too small	4

Relational Algebraic Expression

$\pi_{\text{product . name, complaints . complaint\_text, rating . rating}}$ $\text{Complaints}_{\bowtie \text{complaints.product\_id = rating.product\_id and complaints.customer\_id = rating.customer\_id}}$ Rating , product.product_id = complaints.product_id product

Query 4: Shows the number of orders, number of complaints and average rating for each product

```
select product.name,
(select count(*) from orders where orders.product_id = product.product_id) as
number_of_orders,
(select count(complaints.customer_id) from complaints where
complaints.product_id = product.product_id) as number_of_complaints,
(select avg(rating.rating) from rating where rating.customer_id =
product.product_id) as average_rating
from product
```

Using nested queries is more efficient and accurate than using joins for this query.

name	number_of_orders	number_of_complaints	average_rating
Apple	16	16	4.3333
Banana	13	10	5.4444
Pear	13	12	6.7500
Strawberry	8	9	5.7143
Watermelon	12	8	6.4000
Muskmelon	12	13	7.5000
Grape	12	10	4.7000
Kiwi	16	7	7.8000
Potato	9	7	6.5714
Cucumber	12	10	5.2000
Kale	12	10	5.2000
Lettuce	7	16	5.0000
Apple Juice	15	6	5.0000
Mango Juice	9	13	5.7500

Relational Algebraic Expression

π product.name,(π COUNT (*)

γ COUNT (*) σ orders . product_id = product . product_id orders) → number_of_orders ,

(π COUNT (customer_id)

γ COUNT (customer_id)

σ complaints . product_id = product . product_id complaints) → number_of_complaints,

(π AVG (rating)

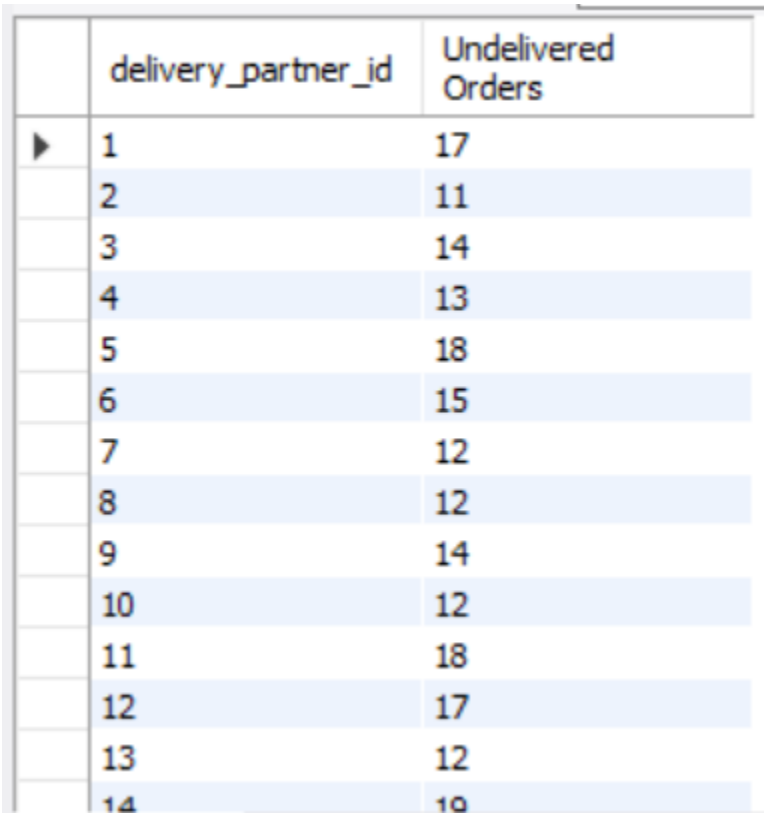
γ AVG (rating)

σ rating . customer_id = product . product_id rating) → average_rating product

Query 5: Lists number of undelivered orders for each delivery partner

```
select delivery_Partner.delivery_partner_id,  (select count(*) from Orders
where orders.delivery_partner_id = delivery_partner.delivery_partner_id
and Orders.status <> "Delivered") as "Undelivered Orders"
```

from Delivery_Partner



	delivery_partner_id	Undelivered Orders
▶	1	17
	2	11
	3	14
	4	13
	5	18
	6	15
	7	12
	8	12
	9	14
	10	12
	11	18
	12	17
	13	12
	14	19

Same query using joins instead: This query is more optimised → **Query Optimization**

```
select delivery_partner.delivery_partner_id, count(*)
from delivery_partner
left join orders
on delivery_partner.delivery_partner_id = orders.delivery_partner_id
where status <> "Delivered"
group by delivery_partner_id
```

Relational Algebraic Expression

π delivery_partner . delivery_partner_id , (π COUNT (*) γ COUNT (*) σ orders . delivery_partner_id = delivery_partner . delivery_partner_id AND orders . status <> "Delivered" orders) →"undelivered orders" delivery_partner

Query 6: Mobile Numbers of delivery partners corresponding to customer ids whos order they are delivering

```
select c.customer_id, d.phone_number
from customer as c, delivery_partner as d, orders as o
where o.customer_id=c.customer_id and o.delivery_partner_id=d.delivery_partner_id
```

customer_id	phone_number
14	2113025126
70	2113025126
92	2113025126
15	2113025126
117	9878598187
55	0150528290
96	0150528290
66	0150528290
115	0150528290
116	0150528290
17	0150528290
4	8618677826
41	8618677826
81	8618677826
79	8618677826
61	8618677826
88	8618677826
114	8618677826
58	8618677826
86	8204566447
58	8204566447
71	8204566447

Relational Algebraic Expression

$$\pi_{c.customer\_id,\ d.phone\_number}\ \sigma_{o.customer\_id = c.customer\_id\ AND\ o.delivery\_partner\_id = d.delivery\_partner\_id}\ (\rho_{c}\,customer \times \rho_{d}\,delivery\_partner \times \rho_{o}\,orders)$$

Query 7: Products listed with their categories, that have the best reviews

```
select product.product_id, product.name, category.name,
(select avg(Rating.rating) from Rating
where Rating.product_id = Product.product_id)
as "Average Rating"
from Product, Category
where 7<(select avg(Rating.rating) from Rating
where Rating.product_id = Product.product_id)
and product.category_id = category.category_id
```

	product_id	name	name	Average Rating
▶	4	Strawberry	Fruit	7.8000
	11	Kale	Vegetable	9.0000
	14	Mango Juice	Juice	9.0000
	19	Curd	Dairy	8.3333
	22	Paneer	Dairy	10.0000
	24	Hide n Seek	Biscuit	8.5000
	26	Parle-G	Biscuit	8.3333

Relational Algebraic Expression

π product . product_id, product . name, category . name, (π AVG (rating)

γ AVG (rating)

σ rating . product_id = product . product_id rating) $\rightarrow$ “Average Rating”

σ product . category_id = category . category_id, 7< (π AVG (rating)

γ AVG (rating)

σ rating . product_id = product . product_id rating)

(product × category)

Query 8: Lists Phone Numbers of Customer, Delivery Partner and Supplier of orders involving Apples

```
select Customer.phone_number as "Customer Phone Number",
Supplier.phone_number as "Supplier Phone Number",
Delivery_Partner.phone_number as "Delivery Partner Phone Number"
from Customer, Supplier, Delivery_Partner, Product, Orders, Supplies
where Product.product_id = Orders.product_id
and Orders.delivery_partner_id = Delivery_Partner.delivery_partner_id
and Orders.customer_id = Customer.customer_id
and Product.product_id = Supplies.product_id
and Supplies.supplier_id = Supplier.supplier_id
and Product.name = "Apple"
```

Customer Phone Number	Supplier Phone Number	Delivery Partner Phone Number
3395277998	9305225037	8618677826
9781326550	9305225037	3301757061
7799796144	9305225037	2113025126
7799796144	9305225037	2571468886
2527358835	9305225037	1854614254
7947966794	9305225037	8618677826
1586876170	9305225037	9731239650
8577396073	9305225037	8614446964
0030080270	9305225037	9731239650
4735353773	9305225037	2374982636
8219685663	9305225037	3301757061

Relational Algebraic Expression

π customer . phone_number, supplier . phone_number, delivery_partner . phone_number

σ product . product_id = orders . product_id AND orders . delivery_partner_id = delivery_partner . delivery_partner_id AND orders . customer_id = customer . customer_id AND product . product_id = supplies . product_id AND supplies . supplier_id = supplier . supplier_id AND product . name = "Apple" (customer × supplier × delivery_partner × product × orders × supplies)

Query 9: supplier who supplies fruits or vegetable

```
select s.supplier_id
from supplier as s,supplies as sup , product as p, category as c
where s.supplier_id=sup.supplier_id and sup.product_id=p.product_id and
p.category_id=c.category_id and c.name="fruits"
UNION
select s.supplier_id
from supplier as s,supplies as sup , product as p, category as c
where s.supplier_id=sup.supplier_id and sup.product_id=p.product_id and
p.category_id=c.category_id and c.name="vegetables"
```

supplier_id
7
8
3
2
1
9
5
6
4

Relational Algebraic Expression

$$\pi_{s.supplier\_id}$$

$$\sigma_{s.supplier\_id = sup.supplier\_id \text{ AND } sup.product\_id = p.product\_id \text{ AND } p.category\_id = c.category\_id \text{ AND } c.name = \text{"fruits"}}(\rho_{s}\,supplier \times \rho_{sup}\,supplies \times \rho_{p}\,product \times \rho_{c}\,category)$$

$$\cup$$

$$\pi_{s.supplier\_id}$$

$$\sigma_{s.supplier\_id = sup.supplier\_id \text{ AND } sup.product\_id = p.product\_id \text{ AND } p.category\_id = c.category\_id \text{ AND } c.name = \text{"vegetables"}}\ (\rho_{s}\,supplier \times \rho_{sup}\,supplies \times \rho_{p}\,product \times \rho_{c}\,category)$$

Query 10: Suppliers who need restocking of their products:

```
Select S.supplier_id, P.name, P.stock
From supplier as S, supplies as sup, products as P
where s.supplier_id=sup.supplier_id
and sup.product_id=P.product_id
 and P.stock<5
```

Relational Algebraic Expression

$$\pi_{s.supplier\_id,\ p.name,\ p.stock}\ \sigma_{s.supplier\_id = sup.supplier\_id\ AND\ sup.product\_id = p.product\_id\ AND\ p.stock < 5}\ (\rho_{s}\ supplier \times \rho_{sup}\ supplies \times \rho_{p}\ products)$$

	supplier_id	name	stock
▶	1	Kiwi	2
	8	Kale	2
	4	Lettuce	4
	7	Milk	4
	6	Yogurt	3
	8	Hide n Seek	2
	5	Bourboun	4
	8	Pork	3

Query 11: Find the second highest price of all products

```
select product.name, max(product.price), category.name
from product
left join category
on product.category_id = category.category_id
where price<(select max(price) from product)
```

	name	max(product.price)	name
▶	Apple	98.00	Fruit

Relational Algebraic Expression

$\pi_{\text{product. name, category . name,max(product.price)}}$

$\sigma_{\text{price}<\pi \text{ MAX (price)}}$ $\gamma_{\text{MAX (price) product}}$

(Category $\ltimes_{\text{product.category\_id = category.category\_id}}$ Product)

Query 12: Find the average price for each category

```
select category.name, avg(product.price)
from product
left join category
on product.category_id = category.category_id
group by category.name
```

	name	avg(product.price)
▶	Fruit	70.375000
	Vegetable	63.500000
	Juice	38.000000
	Dairy	57.333333
	Biscuit	22.250000
	Meat	92.250000

Relational Algebraic Expression

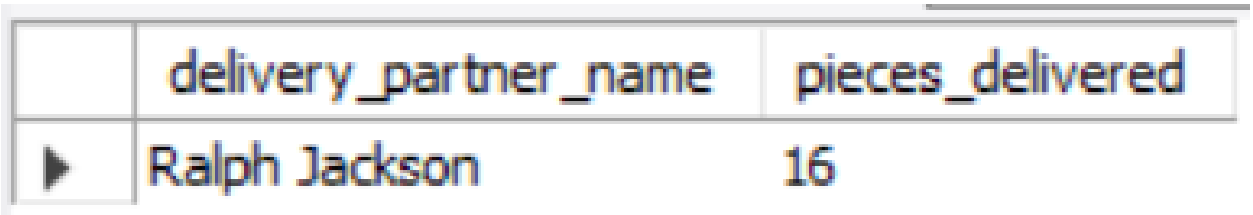
$\pi_{\text{category . name, p . name,avg(product.price)}}$

Category $\ltimes_{\text{product.category\_id = category.category\_id}}$ Product

$\gamma_{\text{category.name}}$ Product

Query 13: Displays the name of the delivery partner who has delivered the maximum number of pieces

```
select CONCAT(delivery_partner.first_name," ",delivery_partner.last_name) as
delivery_partner_name, max(total_quantity) as pieces_delivered
from
(select delivery_partner_id, (sum(quantity)) as total_quantity
from orders
where status = "Delivered"
group by delivery_partner_id) as derived_table
left join delivery_partner
on derived_table.delivery_partner_id = delivery_partner.delivery_partner_id
```



	delivery_partner_name	pieces_delivered
▶	Ralph Jackson	16

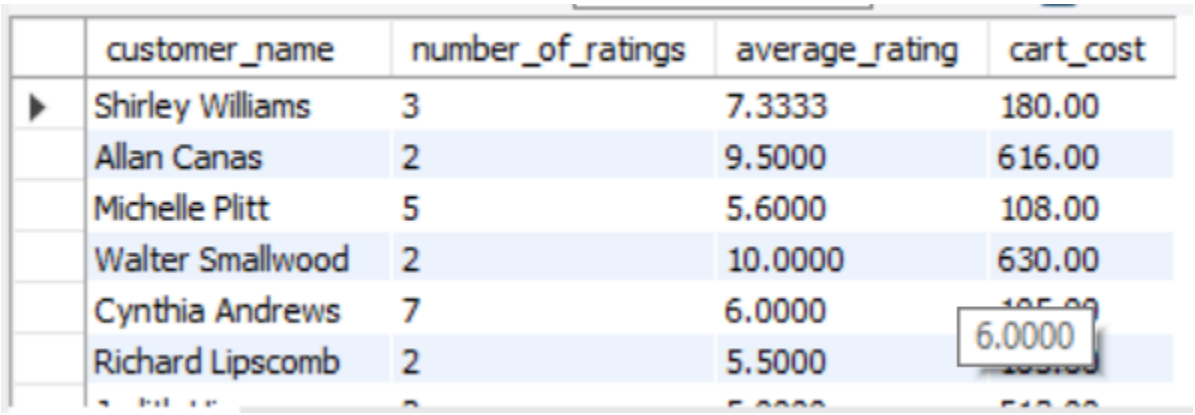
Relational Algebraic Expression

π deliver_partner.first_ name+deliver_partner.last_ name → delivery_partner_name,max(total_quantity) → total quantity
Delivery_partner$_{\propto}$ (π delivery_partner_id, SUM (quantity) → total_quantity
γ delivery_partner_id, SUM (quantity)
σ status = "Delivered" orders) →
Derived_table

Query 14: Shows the number of ratings and average ratings put in by each customer who have a cart cost greater than 0

```
select concat(customer.first_name,' ',customer.last_name) as customer_name,
count(*) as number_of_ratings,
avg(rating.rating) as average_rating, customer.cart_cost
from customer
left join rating
```

on customer.customer_id = rating.customer_id
group by customer.customer_id
having customer.cart_cost > 0



customer_name	number_of_ratings	average_rating	cart_cost
Shirley Williams	3	7.3333	180.00
Allan Canas	2	9.5000	616.00
Michelle Plitt	5	5.6000	108.00
Walter Smallwood	2	10.0000	630.00
Cynthia Andrews	7	6.0000	
Richard Lipscomb	2	5.5000	

Query 15: Returns the address of the customers who have provided bad reviews on products supplied by supplier 1

```
select distinct(CONCAT(c.first_name, ' ', c.last_name)) as customer_name,
c.phone_number,
CONCAT(c.house_number, ', ', c.street, ', ', c.city, ', ', c.state, ', ', c.country) as
Customer_Address,
product.name as Product_name, rating.rating
from supplier
left join (supplies
left join (product
left join (rating
left join customer as c
on c.customer_id = rating.customer_id)
on product.product_id = rating.product_id)
on supplies.product_id = product.product_id)
on supplier.supplier_id = supplies.supplier_id
where rating.rating < 3
and supplier.supplier_id = 1
```

customer_name	phone_number	Customer_Address	Product_name	rating
Kellie Alva	3177375459	65, Sector 9, Mumbai, Maharashtra, India	Apple	2
Abigail Prince	6809102636	374, Ambedkar Nagar, Delhi, Delhi, India	Apple	1
Sophie Bremer	6539714446	661, MG Road, Noida, Uttar Pradesh, India	Apple	2
David Teasley	9878228342	627, Sector 44, Delhi, Delhi, India	Hide n Seek	2
Mark Headley	8262600676	693, Harkesh Nagar, Mumbai, Maharashtra, India	Bourboun	1
Michelle Plitt	7706932988	749, Harkesh Nagar, Noida, Uttar Pradesh, India	Bourboun	2
Ann Orr	3529518072	687, Sector 9, Noida, Uttar Pradesh, India	Bourboun	2

All queries are the most optimised method of doing these tasks. In some queries, multiple versions of the same query are given, and it is specified which is the most optimised form of the query.

EMBEDDED QUERIES->

1)

->For displaying the products

```
req.headers.authorization && db.query(`
        SELECT
            *
        FROM
            product
    `,
```

2)

->For adding products to cart

```
    db.query(`
        INSERT INTO
            cart
        VALUES
            (${req.body.product_id}, ${req.headers.authorization}, 1)
        `,
```

3)

->For displaying all the products that have been added to the cart

```
db.query(`
        SELECT *
        FROM
            cart
```

```
                INNER JOIN
                        product
                ON
                        product.product_id=cart.product_id
                        AND cart.customer_id=${req.headers.authorization}
                `,
4)
->For deleting products from the cart
db.query(`
                DELETE FROM
                        cart
                WHERE
                        customer_id=${req.headers.authorization} AND
                        product_id=${req.params.product_id}
                `,
5)
->For signing up
db.query(
   ` INSERT INTO
      account
      (email, password)
    VALUES
      ("${req.body.email}", "${req.body.password}")
   `,
6)
->For logging in
  db.query(
   ` SELECT
      customer_id
    FROM
      account
    INNER JOIN
      customer
    ON
      account.account_id=customer.account_id
      AND password="${req.body.password}"
      AND email="${req.body.email}"
   `,

7)
```

```
db.query(
      ` INSERT INTO
          customer
          (
            account_id,
            first_name,
            last_name,
            age,
            phone_number,
            house_number,
            street,
            city,
            state,
            country
          )
        VALUES
        (
          ${res2[0].account_id},
          "${req.body.firstName}",
          "${req.body.lastName}",
          ${req.body.age},
          ${req.body.phoneNumber},
          ${req.body.houseNumber},
          "${req.body.street}",
          "${req.body.city}",
          "${req.body.state}",
          "${req.body.country}"
          )
      `,
```

8)FOR RATING THE PRODUCTS->

```
req.headers.authorization && db.query(`
                INSERT INTO
                        rating
                        (product_id, customer_id, rating)
                VALUES
                        (${req.body.product_id}, ${req.headers.authorization},
${req.body.rating})
                ON DUPLICATE KEY UPDATE
                        rating=${req.body.rating}
```

```
    `,
```

9)FOR DELETING THE RATINGS OF PRODUCTS->

```
db.query(`
            DELETE FROM
                rating
            WHERE
                product_id=${req.body.product_id} AND
                customer_id=${req.headers.authorization}
    `,
```

Indexing

Indexes are special lookup tables that the database search engine can use to speed up data retrieval. An Index helps to speed up SELECT queries and WHERE clauses.

Implicit indexes are indexes that are automatically created by the database server when an object is created. Implicit indexes are automatically created for primary key constraints. Thus we do not need to create indexes manually for the following tables: Account, Customer, Supplier, Category, Product, Delivery_Partner, Coupon

Both the Rating and Complaints table use the product_id and the customer_id to retrieve data. So it makes sense to

create indexes on both the tables on product_id, customer_id.

```
create unique index ix1 on rating(product_id,
customer_id)
create unique index ix2 on complaints(product_id,
customer_id)
```

We have added the unique constraint, to make sure that each customer can only add 1 complaint and 1 review for each product.

While these indexes are used automatically when using the select statement on these tables, we can force to use these indexes.

For example, in this query, we have forced the use of the index:

```
select product.name as Product_Name,
concat(customer.first_name, ' ', customer.last_name)
as Customer_Name, rating.rating
from rating use index (ix1), product, customer
where rating.product_id = product.product_id
and customer.customer_id = rating.customer_id
order by rating.rating desc
```

Triggers

To support our database, we have made the following triggers. Also to support the triggers, we have made the following PL SQL procedures:

```
DELIMITER //
create procedure check_stock(IN quantity int, IN p_id
int)
BEGIN
if quantity > (select stock from product where
product_id = p_id)
then SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
'Insufficient Stock';
end if;
end//
DELIMITER ;

DELIMITER //
create procedure reduce_stock(IN quantity int, IN p_id
int)
BEGIN
update product set stock = (select * from (select
stock from product where product_id = p_id) as stck) -
quantity;
end//
DELIMITER ;

DELIMITER //
create procedure check_if_user_coupon(IN c1 int, IN c2
int)
BEGIN
```

```
if (select count(*) from customer_coupon as cc where
cc.customer_id = c1 and cc.coupon_id = c2) = 0
then SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No
such coupon';
end if;
end//
DELIMITER ;

DELIMITER //
create procedure use_up_coupon(IN c1 int, IN c2 int)
BEGIN
delete from Customer_Coupon as cc where cc.customer_id
= c1 and cc.coupon_id = c2;
end//
DELIMITER ;
```

These are the triggers that we have made:

```
create trigger update_cost
after insert
on Cart
for each row
update customer
set cart_cost = (select * from (select
customer.cart_cost from customer where
customer.customer_id=new.customer_id) as cst) +
(new.quantity * (select product.price from product
where product.product_id = new.product_id))
where customer_id = new.customer_id

create trigger stock_limit_cart
before insert
on Cart
for each row
call check_stock(new.product_id, new.quantity);
```

```
create trigger coupon_exists_with_user
before insert
on Orders
for each row
call check_if_user_coupon(new.customer_id,
new.coupon_id);

create trigger coupon_order
after insert
on Orders
for each row
call use_up_coupon(new.customer_id, new.coupon_id);
```

Queries used to create these tables

```
create table Account(
    account_id int PRIMARY KEY,
    email varchar(30) NOT NULL UNIQUE,
    password varchar(30) NOT NULL
    )

create table Customer(
    customer_id int PRIMARY KEY,
    account_id int UNIQUE,
    first_name varchar(30) NOT NULL,
    last_name varchar(30),
    age int,
    phone_number varchar(15),
    house_number varchar(20),
    street varchar(40),
    city varchar(20),
    state varchar(20),
    country varchar(20),
    cart_cost numeric(8,2) DEFAULT 0,

    FOREIGN KEY (account_id) REFERENCES
Account(account_id) ON DELETE CASCADE,
    CHECK (age >= 13)
    )

create table Supplier(
```

```
    supplier_id int PRIMARY KEY,
    account_id int UNIQUE,
    first_name varchar(30) NOT NULL,
    last_name varchar(30),
    age int,
    phone_number varchar(15) NOT NULL,
    house_number varchar(20) NOT NULL,
    street varchar(40) NOT NULL,
    city varchar(20) NOT NULL,
    state varchar(20) NOT NULL,
    country varchar(20) NOT NULL,

    FOREIGN KEY (account_id) REFERENCES
Account(account_id) ON DELETE CASCADE,
    CHECK (age>=18)
    )

create table Category(
    category_id int PRIMARY KEY,
    name varchar(30)
    )

create table Product(
    product_id int PRIMARY KEY,
    name varchar(30) NOT NULL,
    stock int NOT NULL,
    price numeric(8,2) NOT NULL,
    category_id int ,

    FOREIGN KEY (category_id) REFERENCES
Category(category_id)
```

```
    )

create table Delivery_Partner(
    delivery_partner_id int PRIMARY KEY,
    account_id int UNIQUE NOT NULL,
    first_name varchar(30) NOT NULL,
    last_name varchar(30),
    age int,
    phone_number varchar(15) NOT NULL,
    house_number varchar(20) NOT NULL,
    street varchar(40) NOT NULL,
    city varchar(20) NOT NULL,
    state varchar(20) NOT NULL,
    country varchar(20) NOT NULL,

    FOREIGN KEY (account_id) REFERENCES
Account(account_id) ON DELETE CASCADE,
    CHECK (age>=18)
    )


create table Supplies(
    supplier_id int NOT NULL,
    product_id int NOT NULL,

    FOREIGN KEY (supplier_id) REFERENCES
Supplier(supplier_id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES
Product(product_id) ON DELETE CASCADE
    )
```

```
create table Rating(
    product_id int NOT NULL,
    customer_id int NOT NULL,
    rating int NOT NULL,


    FOREIGN KEY (product_id) REFERENCES
Product(product_id) ON DELETE CASCADE,
    FOREIGN KEY (customer_id) REFERENCES
Customer(customer_id) ON DELETE CASCADE,
    CHECK (rating BETWEEN 1 and 11)
    )


create table Complaints(
    product_id int NOT NULL,
    customer_id int NOT NULL,
    complaint_text text NOT NULL,


    FOREIGN KEY (product_id) REFERENCES
Product(product_id) ON DELETE CASCADE,
    FOREIGN KEY (customer_id) REFERENCES
Customer(customer_id) ON DELETE CASCADE
    )

create table Coupon(
    coupon_id int PRIMARY KEY,
    coupon_text varchar(20),
    valid_upto date,
    pct_discount int NOT NULL,
```

```
    CHECK (pct_discount BETWEEN 0 and 100)
    )

create table Customer_Coupon(

    customer_id int NOT NULL,
    coupon_id int NOT NULL,

    FOREIGN KEY (customer_id) REFERENCES
Customer(customer_id) ON DELETE CASCADE,
    FOREIGN KEY (coupon_id) REFERENCES
Coupon(coupon_id) ON DELETE CASCADE
)

create table Cart(
    product_id int NOT NULL,
    customer_id int NOT NULL,
    quantity int NOT NULL,

    FOREIGN KEY (product_id) REFERENCES
Product(product_id) ON DELETE CASCADE,
    FOREIGN KEY (customer_id) REFERENCES
Customer(customer_id) ON DELETE CASCADE,
    CHECK (quantity>0)

)

create table Orders(
    customer_id int NOT NULL,
    product_id int NOT NULL,
```

```
    delivery_partner_id int NOT NULL,
    quantity int NOT NULL,
    order_date date NOT NULL,
    coupon_id int,
    status varchar(20) NOT NULL,

    FOREIGN KEY (customer_id) REFERENCES
Customer(customer_id) ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES
Product(product_id) ON DELETE CASCADE,
    FOREIGN KEY (delivery_partner_id) REFERENCES
Delivery_Partner(delivery_partner_id) ON DELETE
CASCADE,
    FOREIGN KEY (coupon_id) REFERENCES
Coupon(coupon_id) ON DELETE CASCADE,
    CHECK (quantity>0)
    )
```

Tables

Account

	Field	Type	Null	Key	Default	Extra
▶	account_id	int	NO	PRI	NULL	
	email	varchar(30)	NO	UNI	NULL	
	password	varchar(30)	NO		NULL	

	account_id	email	password
▶	1	ntkafktlvac@gmail.com	ifqr
	2	tztuizlfnr@gmail.com	utyn
	3	swzfwadfdhod@gmail.com	qembmlopqvyw
	4	jivktcn@gmail.com	obxybnzooao
	5	smmilgkb@gmail.com	drikewq
	6	jqqwcxrmgiko@gmail.com	nkebsrua
	7	ahhcwxqc@gmail.com	mrnkkwlta
	8	tntsfpfaesy@gmail.com	rbvvjbhnqso
	9	bwnizf@gmail.com	tgeejdqbbhb
	10	ptglphhwe@gmail.com	yztmdfgbks
	11	uzdrkq@gmail.com	civfvvjxf
	12	fmtpndzd@gmail.com	wegedxswmkot
	13	covzduxoif@gmail.com	upftwwkria
	14	vnep@gmail.com	fadhnul
	15	pvwj@gmail.com	eaedz
	16	uzslnskx@gmail.com	xhons

Customer

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	
account_id	int	YES	UNI	NULL	
first_name	varchar(30)	NO		NULL	
last_name	varchar(30)	YES		NULL	
age	int	YES		NULL	
phone_number	varchar(15)	YES		NULL	
house_number	varchar(20)	YES		NULL	
street	varchar(40)	YES		NULL	
city	varchar(20)	YES		NULL	
state	varchar(20)	YES		NULL	
country	varchar(20)	YES		NULL	
cart_cost	decimal(8,2)	YES		0.00	

customer_id	account_id	first_name	last_name	age	phone_number	house_number	street	city	state	country	cart_cost
122	122	Marcia	Adams	44	2609284228	177	MG Road	Pune	Maharashtra	India	0.00
123	123	Patricia	Hess	33	2680819862	310	Ambedkar Nagar	Aligarh	Uttar Pradesh	India	0.00
124	124	Ronald	Cunningh...	58	5523350181	500	Harkesh Nagar	Noida	Uttar Pradesh	India	0.00
125	125	Maria	George	15	3244364888	173	Sector 9	Mumbai	Maharashtra	India	0.00
126	126	Christian	Pinera	14	2322193780	474	Sector 44	Mumbai	Maharashtra	India	0.00
127	127	Elwanda	Fenton	76	6277922193	946	MG Road	Pune	Maharashtra	India	147.00
128	128	Peter	Ward	15	2763796122	128	MG Road	Delhi	Delhi	India	342.00
129	129	Jason	Guzman	64	2272378247	654	Sector 9	Mumbai	Maharashtra	India	0.00
130	130	Eddie	Griffin	13	0112692581	656	Sector 44	Pune	Maharashtra	India	0.00
131	131	John	Gulinson	73	7237926892	453	Ambedkar Nagar	Pune	Maharashtra	India	0.00
132	132	Eric	Johnson	15	7927915375	268	Ambedkar Nagar	Noida	Uttar Pradesh	India	0.00
133	133	Pamela	Howard	35	9622808477	816	Sector 44	Mumbai	Maharashtra	India	0.00
134	134	Sharon	Morton	22	2874472909	881	Sector 137	Aligarh	Uttar Pradesh	India	0.00
135	135	Ann	Orr	64	3529518072	687	Sector 9	Noida	Uttar Pradesh	India	72.00
136	136	Arthur	Maraldo	49	0181634616	180	Harkesh Nagar	Delhi	Delhi	India	0.00
137	137	Lena	Irland	25	6443023339	559	Ambedkar Nagar	Mumbai	Maharashtra	India	0.00

Supplier

	Field	Type	Null	Key	Default	Extra
▶	supplier_id	int	NO	PRI	NULL	
	account_id	int	YES	UNI	NULL	
	first_name	varchar(30)	NO		NULL	
	last_name	varchar(30)	YES		NULL	
	age	int	YES		NULL	
	phone_number	varchar(15)	NO		NULL	
	house_number	varchar(20)	NO		NULL	
	street	varchar(40)	NO		NULL	
	city	varchar(20)	NO		NULL	
	state	varchar(20)	NO		NULL	
	country	varchar(20)	NO		NULL	

	supplier_id	account_id	first_name	last_name	age	phone_number	house_number	street	city	state	country
▶	1	141	Gregory	Jackson	63	1886370846	944	Harkesh Nagar	Mumbai	Maharashtra	India
	2	142	Rodrigo	Catalano	36	7656719330	327	Sector 44	Noida	Uttar Pradesh	India
	3	143	Greta	Stadler	84	6654372157	141	Sector 44	Noida	Uttar Pradesh	India
	4	144	Sandra	Snowden	62	0494921607	33	Sector 9	Aligarh	Uttar Pradesh	India
	5	145	Samuel	Rosen	85	3627913689	644	Sector 44	Aligarh	Uttar Pradesh	India
	6	146	Darrin	Hull	50	4275438518	368	Sector 137	Aligarh	Uttar Pradesh	India
	7	147	Tony	Kellum	61	8115556440	581	Harkesh Nagar	Delhi	Delhi	India
	8	148	Kevin	Moore	19	8400354958	773	Ambedkar Nagar	Pune	Maharashtra	India
	9	149	Kimberley	Sanderlin	72	3131194710	574	Sector 9	Delhi	Delhi	India
	10	150	Sibyl	Horvath	27	7391679367	13	Sector 9	Delhi	Delhi	India

Category

	Field	Type	Null	Key	Default	Extra
▶	category_id	int	NO	PRI	NULL	
	name	varchar(30)	YES		NULL	

	category_id	name
▶	1	Fruit
	2	Vegetable
	3	Juice
	4	Dairy
	5	Biscuit
	6	Meat

Product

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	NULL	
name	varchar(30)	NO		NULL	
stock	int	NO		NULL	
price	decimal(8,2)	NO		NULL	
category_id	int	YES	MUL	NULL	

product_id	name	stock	price	category_id
1	Apple	15	93.00	1
2	Banana	6	52.00	1
3	Pear	3	57.00	1
4	Strawberry	5	98.00	1
5	Watermelon	5	88.00	1
6	Muskmelon	17	65.00	1
7	Grape	8	71.00	1
8	Kiwi	17	39.00	1
9	Potato	20	50.00	2
10	Cucumber	13	80.00	2
11	Kale	6	57.00	2
12	Lettuce	13	67.00	2
13	Apple Juice	17	32.00	3
14	Mango Juice	16	41.00	3
15	Strawberry...	20	31.00	3
16	Mixed Fruit...	19	48.00	3

Delivery Partner

	Field	Type	Null	Key	Default	Extra
▶	delivery_partner_id	int	NO	PRI	NULL	
	account_id	int	NO	UNI	NULL	
	first_name	varchar(30)	NO		NULL	
	last_name	varchar(30)	YES		NULL	
	age	int	YES		NULL	
	phone_number	varchar(15)	NO		NULL	
	house_number	varchar(20)	NO		NULL	
	street	varchar(40)	NO		NULL	
	city	varchar(20)	NO		NULL	
	state	varchar(20)	NO		NULL	
	country	varchar(20)	NO		NULL	

	delivery_partner_id	account_id	first_name	last_name	age	phone_number	house_number	street	city	state	country
▶	1	151	Ralph	Jackson	38	6860251011	419	Harkesh Nagar	Aligarh	Uttar Pradesh	India
	2	152	Robert	Rogers	71	9114931611	120	Ambedkar Nagar	Noida	Uttar Pradesh	India
	3	153	Ricardo	Jefferson	21	1103878512	693	MG Road	Mumbai	Maharashtra	India
	4	154	Connie	Gregory	35	9685832429	747	Ambedkar Nagar	Pune	Maharashtra	India
	5	155	Bonnie	Huttar	63	0880887382	498	MG Road	Pune	Maharashtra	India
	6	156	Frank	Merrell	63	3392326262	232	Ambedkar Nagar	Aligarh	Uttar Pradesh	India
	7	157	Manuela	Sutton	32	9767250730	363	Govindpuri	Aligarh	Uttar Pradesh	India
	8	158	Bonnie	Sheldon	49	4141026233	932	Ambedkar Nagar	Aligarh	Uttar Pradesh	India
	9	159	Bethany	Jeffries	64	4241752940	266	Ambedkar Nagar	Delhi	Delhi	India
	10	160	Diana	Morris	78	4656531994	751	Ambedkar Nagar	Delhi	Delhi	India
	11	161	Mary	Bleakley	28	6731133735	363	MG Road	Delhi	Delhi	India
	12	162	Bruce	Shuck	47	8995628508	480	Ambedkar Nagar	Pune	Maharashtra	India
	13	163	Irene	Saunders	74	0057032054	532	Sector 9	Aligarh	Uttar Pradesh	India
	14	164	Sandra	Tant	52	2876689643	238	Sector 137	Pune	Maharashtra	India
	15	165	Larry	Darden	60	2281417973	94	Sector 44	Delhi	Delhi	India
	16	166	David	Friedman	71	1870870504	140	Harkesh Nagar	Noida	Uttar Pradesh	India

Supplies

	Field	Type	Null	Key	Default	Extra
▶	supplier_id	int	NO	MUL	NULL	
	product_id	int	NO	MUL	NULL	

	supplier_id	product_id
▶	1	1
	1	2
	9	3
	9	4
	7	5
	4	6
	8	7
	7	8
	8	9
	8	10
	9	11
	2	12
	3	13
	9	14
	8	15
	8	16

Rating

	Field	Type	Null	Key	Default	Extra
▶	product_id	int	NO	MUL	NULL	
	customer_id	int	NO	MUL	NULL	
	rating	int	NO		NULL	

	product_id	customer_id	rating
▶	29	116	8
	2	46	1
	17	14	2
	10	80	7
	12	28	2
	14	70	10
	15	95	5
	17	132	10
	19	10	7
	12	62	1
	20	25	7
	4	58	5
	15	127	3
	1	6	6
	24	59	4
	22	10	1

Complaints

	Field	Type	Null	Key	Default	Extra
▶	product_id	int	NO	MUL	NULL	
	customer_id	int	NO	MUL	NULL	
	complaint_text	text	NO		NULL	

product_id	customer_id	complaint_text
2	92	Bad Colour
11	91	Me do not likey
12	25	Bad Colour
10	17	Broken
1	63	Broken
27	46	Broken
14	45	Bad Picture
20	11	Very Bad
22	117	supplier sent hitmen to my house
25	27	Bad Picture
2	16	Too small
12	72	Bad Colour
1	5	Bad Colour
21	8	Bad
26	73	Me do not likey

Coupon

Field	Type	Null	Key	Default	Extra
coupon_id	int	NO	PRI	NULL	
coupon_text	varchar(20)	YES		NULL	
valid_upto	date	YES		NULL	
pct_discount	int	NO		NULL	

coupon_id	coupon_text	valid_upto	pct_discount
1	SAVING10	2022-08-10	10
2	BACHAT20	2022-07-23	20
3	SUPER25	2022-08-15	25
4	DHAMAAL30	2022-05-03	30
5	DISCOUNT40	2022-07-00	40
6	SAVING50	2022-05-09	50
7	SAVE70	2022-08-11	70

Customer Coupon

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	MUL	NULL	
coupon_id	int	NO	MUL	NULL	

customer_id	coupon_id
36	2
92	
52	7
45	1
117	7
54	5
40	4
101	5
137	1
24	5
112	7
48	2
88	6
114	3
80	4
68	4

Cart

Field	Type	Null	Key	Default	Extra
product_id	int	NO	MUL	NULL	
customer_id	int	NO	MUL	NULL	
quantity	int	NO		NULL	

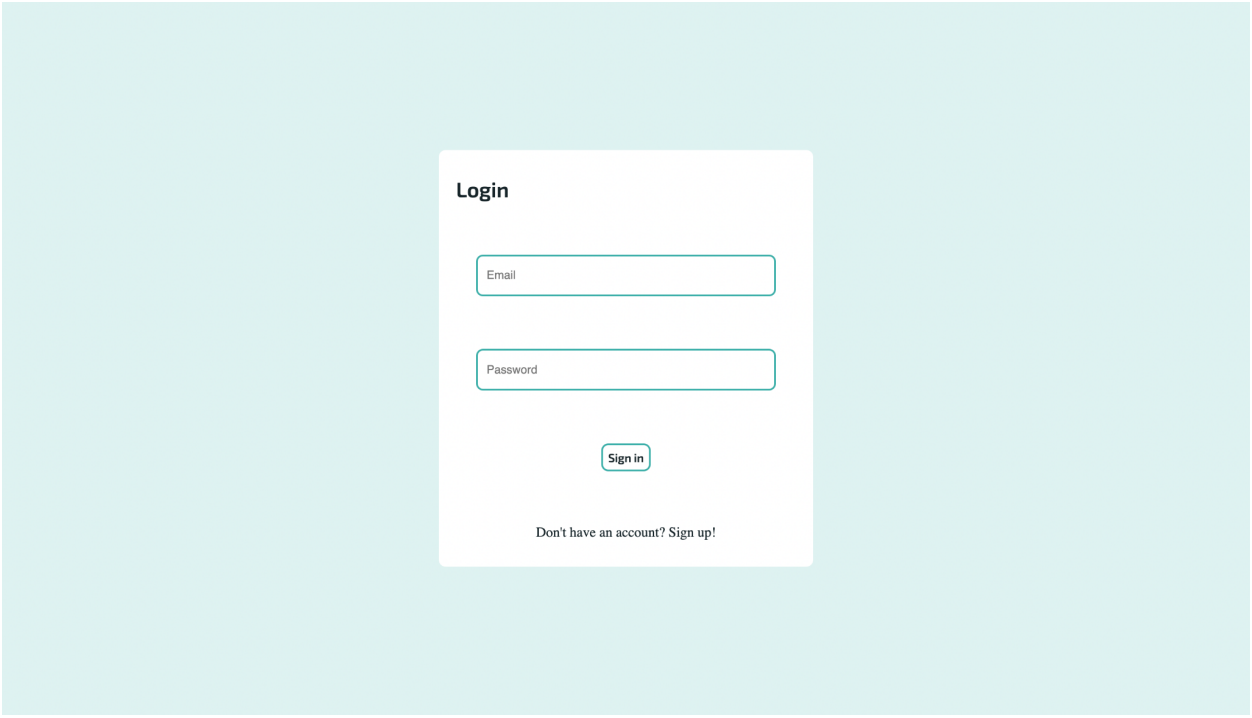
	product_id	customer_id	quantity
▶	5	32	7
	20	14	5
	4	109	7
	22	40	7
	20	39	3
	24	127	7
	20	135	2
	24	53	5
	24	112	6
	3	128	6
	3	78	4
	24	49	5
	3	78	5

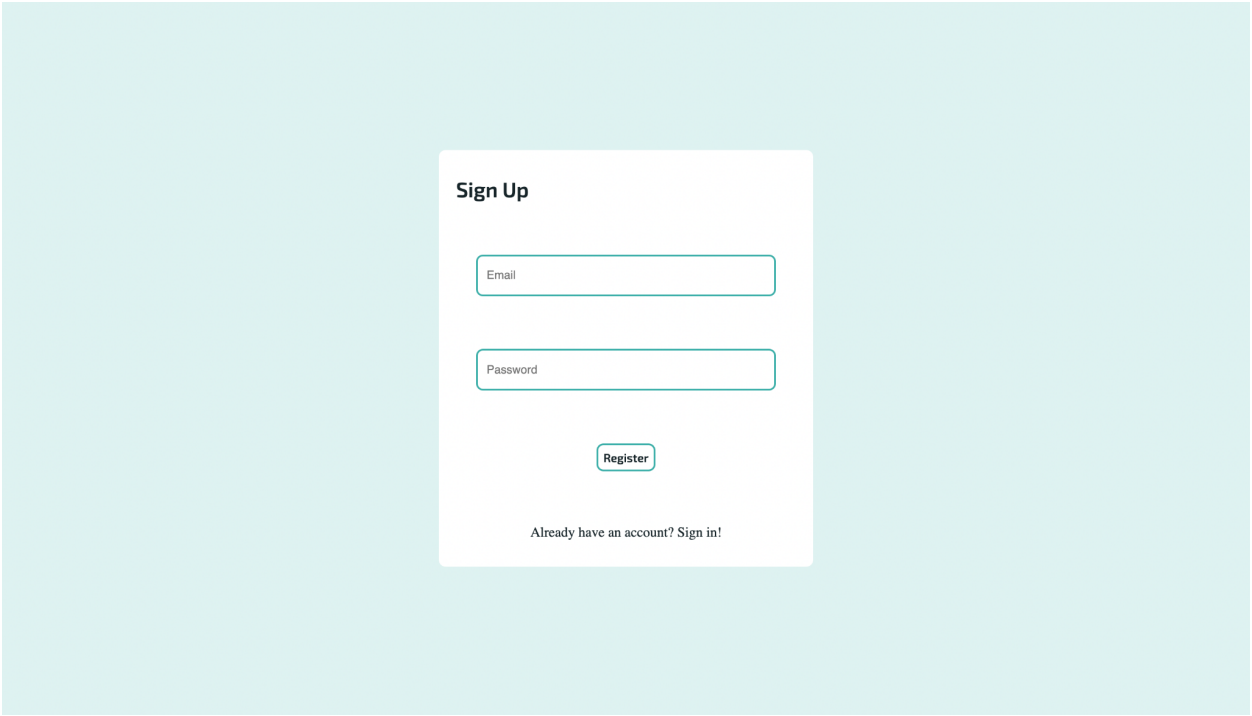
Orders

	Field	Type	Null	Key	Default	Extra
▶	customer_id	int	NO	MUL	NULL	
	product_id	int	NO	MUL	NULL	
	delivery_partner_id	int	NO	MUL	NULL	
	quantity	int	NO		NULL	
	order_date	date	NO		NULL	
	coupon_id	int	YES	MUL	NULL	
	status	varchar(20)	NO		NULL	

customer_id	product_id	delivery_partner_id	quantity	order_date	coupon_id	status
91	20	5	3	2022-04-06	6	Packed
53	8	8	1	2022-04-23	7	Shipped
12	27	18	2	2022-04-13	5	Shipped
84	30	9	1	2022-04-28	5	Shipped
10	18	9	3	2022-03-13	7	Packed
104	5	16	3	2022-03-06	1	Delivered
89	1	5	1	2022-03-22	3	Delivered
108	14	1	3	2022-04-27	3	Processing
32	7	17	1	2022-02-07	5	Shipped
33	13	17	1	2022-04-23	2	Shipped

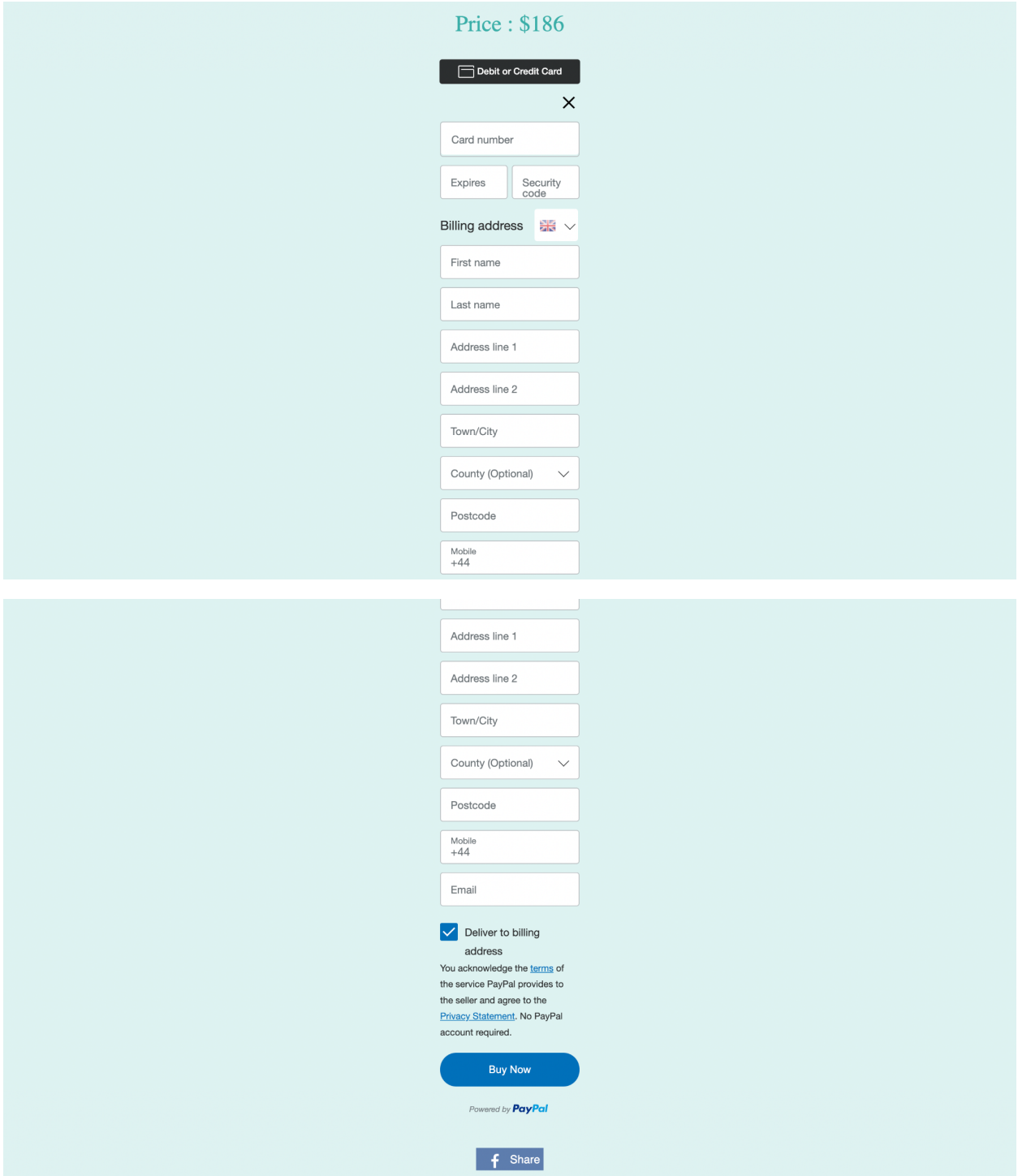
WEBSITE SCREENSHOTS->



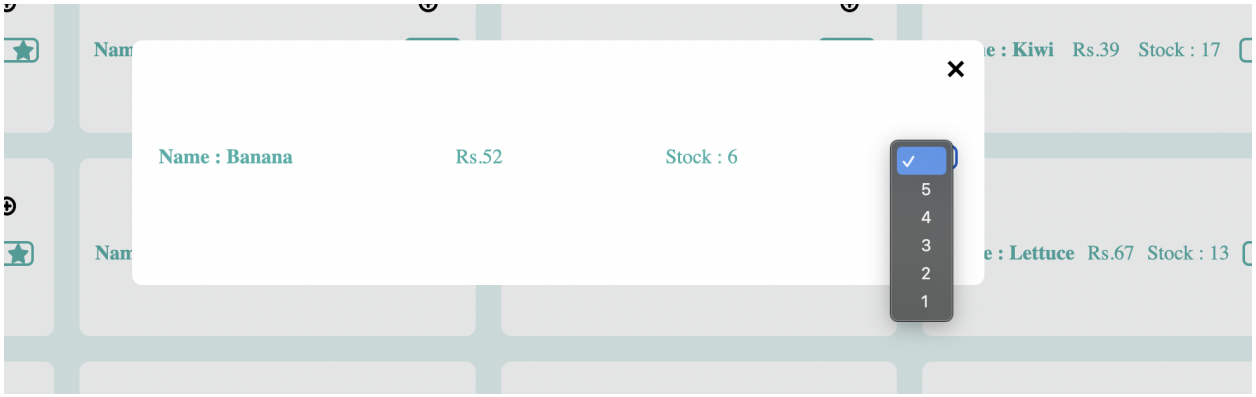


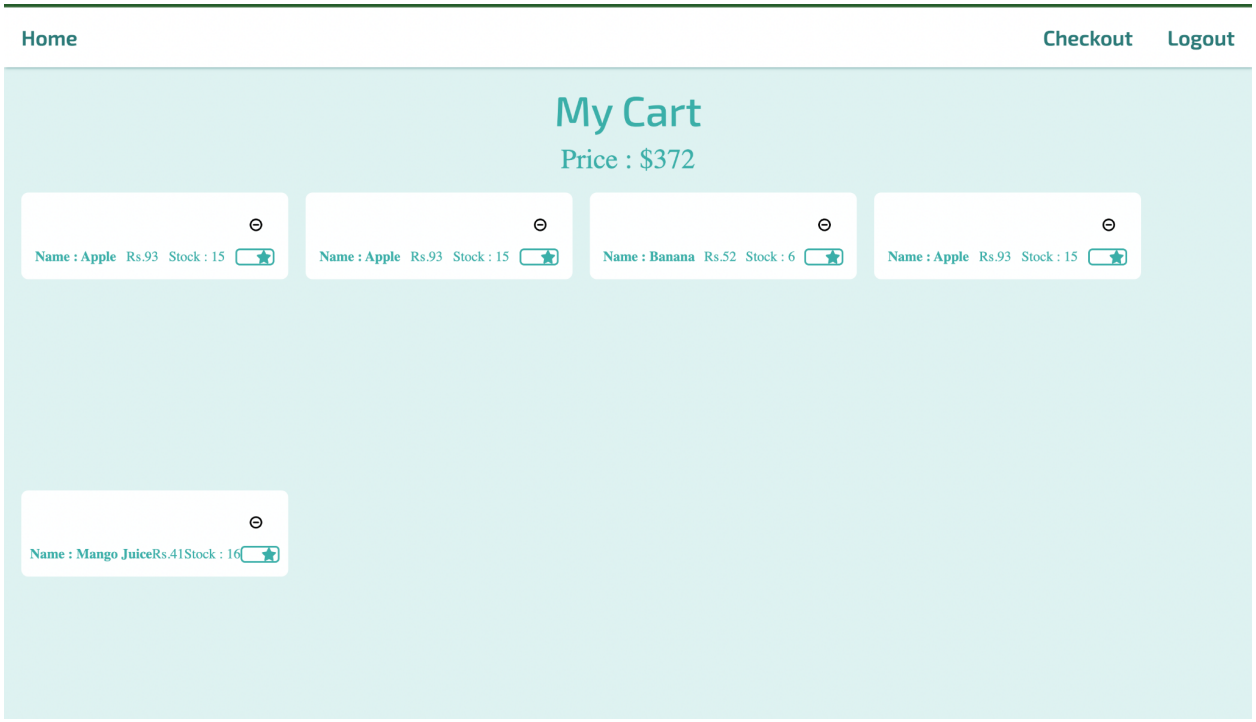
Price : $186

Price : $186


Debit or Credit Card
Card number
Expires
Security code
Billing address
First name
Last name
Address line 1
Address line 2
Town/City
County (Optional)
Postcode
Mobile
+44
Address line 1
Address line 2
Town/City
County (Optional)
Postcode
Mobile
+44
Email
Deliver to billing address
You acknowledge the terms of the service PayPal provides to the seller and agree to the Privacy Statement. No PayPal account required.
Buy Now
Powered by PayPal
Share


Home
Cart
Logout
Name : Apple Rs.93 Stock : 15 5
Name : Banana Rs.52 Stock : 6 5
Name : Pear Rs.57 Stock : 3
Name : Strawberry Rs.98 Stock : 5
Name : Watermelon Rs.88 Stock : 5
Name : Muskmelon Rs.65 Stock : 17
Name : Grape Rs.71 Stock : 8
Name : Kiwi Rs.39 Stock : 17
Name : Potato Rs.50 Stock : 20
Name : Cucumber Rs.80 Stock : 13
Name : Kale Rs.57 Stock : 6
Name : Lettuce Rs.67 Stock : 13
Name : Apple Juice Rs.32 Stock : 17
Name : Mango Juice Rs.41 Stock : 16
Name : Strawberry Juice Rs.31 Stock : 20
Name : Mixed Fruit Juice Rs.48 Stock : 19
Name : Milk Rs.54 Stock : 10
Name : Cheese Rs.73 Stock : 13
Name : Curd Rs.18 Stock : 14
Name : Greek Yogurt Rs.36 Stock : 2


Name : Banana
Rs.52
Stock : 6
5
4
3
2
1
e : Kiwi Rs.39 Stock : 17
e : Lettuce Rs.67 Stock : 13


Home
Checkout
Logout
My Cart
Price : $372
Name : Apple Rs.93 Stock : 15
Name : Apple Rs.93 Stock : 15
Name : Banana Rs.52 Stock : 6
Name : Apple Rs.93 Stock : 15
Name : Mango JuiceRs.41Stock : 16