

Database and Management Systems

End Semester Project Evaluation

Aaryendra Chhabra (2020005)

Aayush Gakhar (2020006)

Ayush Sharma (2020042)

Jayan Pahuja (2020071)

Scope of the project:

Since Covid-19 struck the world, most businesses, jobs and services have gone online. Even though we can say that we've successfully tackled this Covid hurdle today, we won't be able to go back to what we were before the pandemic. Keeping this in mind, we are developing an online retail store service where people can browse and order products they need. The billing and payment will be made electronically. The prices will be fixed with the only exception of applying coupons to reduce the cost of the order and therefore cutting the need for a third person hence decreasing the risk of theft/black marketing. It also keeps track of all the customers as well as suppliers along with the shippers. The suppliers are basically the sellers who are listing their products on this service system with the help of an admin. The admin manages the entire retail store. It can change the listed prices of the products and add or remove coupons, and many more controls. The shippers are the ones who deliver the products from the suppliers to the customers in a given expected interval of time. This system keeps track of all the orders ordered by the customer and products that the sellers/shipping are selling/shipping as well as the admins.

Stakeholder:

Upon conducting some informal research and discussions by the team members, we have identified some of the stakeholders/ target audience of the project as:

- People who like to shop online
- People who want to buy in bulk
- Big and Small Businessmen (Suppliers/ Sellers)
- Shipping/ Transportation service providers
- Management Staff (Admin Controls)

Features to various Stakeholders:

Our project acts as an interface between the customers and the suppliers. The login page consists of three types of users and the grants to each of them have been provided separately. The entities and what functions each of these possess have been mentioned below:

- Customer:
 - ❖ A customer can sign up feature for new account where they can provide their personal information and create a new account for themselves.
 - ❖ Selecting products to be added to cart.
 - ❖ A customer can apply coupons (if possible) to avail discount.
 - ❖ A customer can choose to update the information provided and also add multiple addresses so as to get products at different locations or order for someone else.
 - ❖ A customer can add/delete reviews for each product.
 - ❖ A customer can view products based on categories as well.
 - ❖ A customer can choose to remove items from the cart and not proceed with the order.
 - ❖ A customer can pay using different payment methods to suit needs.
 - ❖ A customer can view the previous orders placed by him/her.
- Employee:
 - ❖ An employee can add new products to the list belonging to different categories.
 - ❖ An employee can delete the existing products, deletion can occur using product name, product ID or product category.

- ❖ An employee can update the fields of a particular product such as product name, product quantity, product price, product weight, product discount and product category ID.
- ❖ An employee can view the information about customers, suppliers, shippers and products in the form of a table.
- ❖ An employee can view the orders placed by different customers from a particular start date to a particular end date or can also search by customer ID.
- ❖ An employee can update the values of different entities like add coupons to the pre-existing ones, add a new category, change information about a particular supplier, change a supplier related to a particular product, add a new shipper, etc.
- Administrator:
 - ❖ An administrator can perform all the above-mentioned roles done by the employee.
 - ❖ An administrator can add new employees to the organisation.
 - ❖ An administrator can update information about other employees and also give admin access to pre-existing employees.
 - ❖ An administrator can remove the employees which have left the company.

Some Assumptions:

- ❖ When a customer adds products to a cart, the coupon will only be valid after a certain number of items selected (a threshold of transaction amount for a purchase).
- ❖ When an administrator/employee places an order for supplies, it is assumed that the quantity of product is automatically increased.

E-R Diagram:

Link to E-R Diagram:

https://lucid.app/lucidchart/24105b34-9ca0-456a-ac8c-06bceac62cf5/edit?invitationId=inv_ace2a9ab-b45c-4a2f-8c9b-b8344eff0864

Entities:

- 1) Customer
- 2) Product
- 3) Supplier
- 4) Category
- 5) Administrator
- 6) Cart
- 7) Orders
- 8) Coupons
- 9) Shipper

Weak Entities:

- 1) Order Details: Order details doesn't exist without placing the Order. Therefore, Order Details is dependent on Order. Hence, it's a weak entity.
- 2) Bills: Bills can't be generated without placing the order. Therefore, Bills can't be generated without placing an Order. Hence, it's a weak entity.
- 3) Personal Information: Personal Information only exists when a Customer exists. Hence, it's a weak entity.
- 4) Supplier Information: Supplier Information only exists when a Supplier exists. Hence, it's a weak entity.
- 5) Review: A review can only be given on a product. Therefore, it depends on the existence of a product. Hence, it's a weak entity.
- 6) Payment: Payment cannot be made until an order has been placed. Hence, it's a weak entity.

Relationships:

- 1) Customer has Personal Information
- 2) Customer places Orders
- 3) Customer adds Products to cart
- 4) Order has Order Details
- 5) Order has Bill
- 6) Cart shows Coupons
- 7) Administrator adds product
- 8) Product belongs to Category
- 9) Supplier supplies product
- 10) Supplier has Personal Information
- 11) Order shipped by Shipper

Ternary Relationship:

- 1) Customer adds Products to cart

-> A binary relationship would not be able to accurately fulfil the semantics of the relationship between the Customer, Product and Cart.

-> As all of the above entities are inter-connected to each other, therefore a ternary relationship is appropriate in this case.

Relational Schema along with primary keys and valid constraints:

Link:

<https://docs.google.com/spreadsheets/d/1shdIGC4zyjXVXN4HNL4FNLNrs8IhiD1h5IPBaOEfIs/edit?usp=sharing>

The primary and foreign keys for different tables are mentioned below:

- ❖ Customer:
(Primary Key – Customer_ID)
- ❖ Customer Information:
(Foreign Key – Customer_ID)
- ❖ Supplier:
(Primary Key – Supplier_ID)
- ❖ Supplier Information:
(Foreign Key – Supplier_ID)
- ❖ Product:
(Primary Key – Product_ID), (Foreign Key – Category_ID)
- ❖ Category:
(Primary Key – Category_ID)
- ❖ Shipper:
(Primary Key – Shipper_ID)
- ❖ Employee:
(Primary Key – Employee_ID)

(ON DELETE CASCADE has been used to cascade the entries when some attribute which is a foreign key for a particular table has been deleted from its original table.)

The constraints added to various Tables are mentioned below:

Tables:

The tables which have been used are mentioned as follows:

Tables_in_retailstore

- ❖ 'adds_product'
- ❖ 'administrator'
- ❖ 'applied_coup'
- ❖ 'bills'
- ❖ 'cart'
- ❖ 'cart_details'
- ❖ 'category'
- ❖ 'coupons'
- ❖ 'customer'
- ❖ 'customer_details'
- ❖ 'customer_information'

- ❖ 'employee'
- ❖ 'order_details'
- ❖ 'orders'
- ❖ 'payment'
- ❖ 'places_order'
- ❖ 'product'
- ❖ 'product_1'
- ❖ 'product_2'
- ❖ 'product_details'
- ❖ 'product_view'
- ❖ 'products_categoric'
- ❖ 'review'
- ❖ 'shipper'
- ❖ 'shipper_details'
- ❖ 'supplier'
- ❖ 'supplier_details'
- ❖ 'supplier_information'

These tables also contain views.

Indexes:

- ❖ CREATE INDEX customer_id_index
ON customer(cust_id);
- ❖ CREATE INDEX product_id_index
ON product(product_id);
- ❖ CREATE INDEX product_category_id_index
ON product(category_id);
- ❖ CREATE INDEX product_id_index
ON product(product_id);
- ❖ CREATE INDEX order_id_index
ON orders(order_id);
- ❖ CREATE INDEX order_id_index
ON orders(customer_id);
- ❖ CREATE INDEX order_details_index
ON order_details(customer_id);

- ❖ CREATE INDEX supplier_id_index
ON supplier(supp_id);
- ❖ CREATE INDEX shipper_id_index
ON shipper(ship_id);

Range Constraints:

Some constraints used on the fields are as follows:

- ```
CREATE TABLE SUPPLIER(
 SUPP_ID CHAR(4) PRIMARY KEY,
 EMAIL_ID VARCHAR(50) NOT NULL,
 COMPANY_NAME VARCHAR(50) NOT NULL,
 CHECK (SUPP_ID LIKE 'S%'),
 CHECK (EMAIL_ID LIKE '%@%.%'));
```
- ❖
- ```
CREATE TABLE CUSTOMER(
  CUST_ID CHAR(4) PRIMARY KEY,
  EMAIL_ID VARCHAR(50) NOT NULL ,
  FIRST_NAME VARCHAR(20) NOT NULL,
  LAST_NAME VARCHAR(20) NOT NULL,
  C_PASSWORD VARCHAR(30) NOT NULL,
  CHECK (EMAIL_ID LIKE '%@%.%'),
  CHECK (LENGTH(C_PASSWORD)>1 ),
  CHECK (CUST_ID LIKE 'C%'));
```
- ❖
- ```
DISCOUNT INTEGER NOT NULL,
CATEGORY_ID CHAR(5),
IN_STOCK BOOLEAN NOT NULL,
CHECK (PRODUCT_ID LIKE 'P%'),
CHECK (PRODUCT_QUANTITY > 0),
CHECK (UNIT_WEIGHT > 0),
CHECK (UNIT_PRICE > 0));
```
- ❖
- ```
EMAIL_ID VARCHAR(50) NOT NULL,
A_PASSWORD VARCHAR(30) NOT NULL,
ADMIN_ROLES BOOLEAN not NULL,
CHECK (ADMIN_ID LIKE 'E%'),
CHECK (LENGTH(A_PASSWORD)>1 ));
```
- ❖
- ```
TOTAL_QUANTITY INTEGER NOT NULL,
COUPON_APPLIED BOOLEAN,
CHECK (TOTAL_PRICE >= 0 AND TOTAL_QUANTITY >= 0),
CHECK (CART_ID LIKE 'CA%'));
```
- ❖
- ```
QUANTITY INTEGER NOT NULL,
DISCOUNT INTEGER NOT NULL,
CHECK (UNIT_PRICE > 0 AND QUANTITY > 0 AND DISCOUNT >= 0));
```
- ❖

SQL Queries:

- ❖ SELECT distinct *
FROM Supplier_information,supplier
Where supplier_information.personal_id = supplier.supp_id;

| | PERSONAL_ID | LOCALITY | CITY | STATE | COUNTRY | POSTAL_CODE | PHONE_NUMBER | SUPP_ID | EMAIL_ID |
|---|-------------|---------------------------|---------------|----------------------|---------------|-------------|--------------|---------|-------------------------|
| ▶ | S100 | 748 Algoma Crossing | Washington | District of Columbia | United States | 20210 | 2025627387 | S100 | vivamus.nibh@google. |
| | S101 | 59978 Blaine Parkway | Santa Fe | New Mexico | United States | 87592 | 5051838969 | S101 | amet@yahoo.org |
| | S102 | 44 Twin Pines Pass | Lubbock | Texas | United States | 79405 | 8063798402 | S102 | mauris@yahoo.co.uk |
| | S103 | 811 Barnett Way | Plano | Texas | United States | 75074 | 4696165874 | S103 | faucibus@protonmail.c |
| | S104 | 6 New Castle Junction | Bradenton | Florida | United States | 34282 | 9415390592 | S104 | ut.odio@yahoo.org |
| | S105 | 3386 Susan Court | Fresno | California | United States | 93762 | 5595445302 | S105 | donec.fringilla.donec@ |
| | S106 | 8679 Bultman Park | Washington | District of Columbia | United States | 20337 | 2027748968 | S106 | sollicitudin.commodo.ip |
| | S107 | 7338 Scoville Trail | Philadelphia | Pennsylvania | United States | 19146 | 2672785872 | S107 | tempus.risus@outlook. |
| | S108 | 05 Dawn Crossing | Laredo | Texas | United States | 78044 | 9562685217 | S108 | tempor.augue@proton |
| | S109 | 84530 Brickson Park Point | Atlanta | Georgia | United States | 31136 | 4049011429 | S109 | cras.dictum@yahoo.ne |
| | S110 | 35609 Briar Crest Pass | Albuquerque | New Mexico | United States | 87121 | 5057410368 | S110 | et.euismod.et@yahoo |
| | S111 | 41751 Kim Plaza | Fort Worth | Texas | United States | 76147 | 8175194021 | S111 | aliquet.diam@yahoo.e |
| | S112 | 51 Barby Drive | Colorado S... | Colorado | United States | 80940 | 7194463445 | S112 | id.sapien.cras@icloud. |
| | S113 | 8769 8th Point | Myrtle Beach | South Carolina | United States | 29579 | 8431585241 | S113 | malesuada.malesuada |
| | S114 | 86802 Packers Circle | Charlotte | North Carolina | United States | 28247 | 7046193715 | S114 | urna.nec.luctus@outlo |
| | S115 | 3420 Marnheron Pass | Dayton | Ohio | United States | 45419 | 9373478951 | S115 | arcu.aliquam ultrices@ |

- ❖ Select Cust_ID
From Customer join Customer_Information
Where Customer_Information.state = "Washington" and Cust_ID =
Customer_Information.Personal_ID;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

| | Cust_ID |
|---|---------|
| ▶ | C114 |
| | C155 |
| | C120 |

- ❖ Select product_id
From product
Where discount > 10;
- ❖ Select category_name, count(product_id)
From category join product
Where category.category_id = product.category_id
Group By category_name;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

| | category_name | count(product_id) |
|---|---------------|-------------------|
| ▶ | COSMETICS | 15 |
| | STATIONERY | 15 |
| | ELECTRONICS | 15 |
| | PHARMACY | 15 |
| | TOYS | 15 |
| | FURNITURE | 15 |
| | CLOTHES | 15 |
| | SHOES | 15 |
| | GROCERY | 15 |
| | SPORTS | 15 |

- ❖ Select product.product_name from order_details,product
Where order_details.order_id = 'o100' and product.product_id =
order_details.product_id;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

| | product_name |
|---|--------------|
| ▶ | grocery10 |
| | clothes02 |
| | furniture11 |
| | grocery06 |
| | grocery03 |
| | stationery07 |
| | furniture05 |
| | sports02 |
| | pharmacy02 |
| | toys09 |
| | grocery02 |

- ❖ Select product_id,avg(rating) as average_rating
From review
Group by product_id;

| product_id | average_rating |
|------------|----------------|
| ▶ P121 | 2.4286 |
| P125 | 3.6000 |
| P206 | 3.8333 |
| P160 | 3.0000 |
| P154 | 2.2500 |
| P214 | 2.6667 |
| P105 | 3.0000 |
| P238 | 3.2222 |
| P236 | 2.8333 |
| P159 | 2.2000 |
| P132 | 4.0000 |
| P230 | 3.0000 |
| P106 | 2.7500 |
| P124 | 3.5000 |

- ❖ ALTER TABLE ORDERS
ADD FOREIGN KEY (BILL_ID) REFERENCES BILLS(BILL_ID) ON DELETE
CASCADE;
- ❖ CREATE TABLE ORDERS(
ORDER_ID CHAR(4) PRIMARY KEY,
CUST_ID CHAR(4),
SUPP_ID CHAR(4),
BILL_ID CHAR(4),
SHIPPING_ID CHAR(5),
CONSIGNMENT_NUMBER CHAR(5) NOT NULL,
SHIPPED_DATE DATE NOT NULL,
ORDER_DATE DATE NOT NULL,
FOREIGN KEY (CUST_ID) REFERENCES CUSTOMER(CUST_ID) ON DELETE
CASCADE,
FOREIGN KEY (SUPP_ID) REFERENCES SUPPLIER(SUPP_ID) ON DELETE
CASCADE,
FOREIGN KEY (SHIPPING_ID) REFERENCES SHIPPER(SHIP_ID) ON DELETE
CASCADE,
CHECK (CONSIGNMENT_NUMBER LIKE 'CW%'));
- ❖ SELECT CUST_ID,SUM(PAYMENT.AMOUNT_PAID)
FROM PAYMENT
GROUP BY CUST_ID

| | CUST_ID | SUM(PAYMENT.AMOUNT_PAID) |
|---|---------|--------------------------|
| ▶ | C100 | 912 |
| | C101 | 456 |
| | C102 | 1873 |
| | C103 | 2182 |
| | C104 | 2218 |
| | C105 | 2029 |
| | C106 | 783 |

- ❖ INSERT INTO product values('P149', 'Turkey ', 100, 31, 112, 9, 'CT100');
- ❖ SELECT DISTINCT CA.PRODUCT_ID, PRODUCT_NAME, CA.UNIT_PRICE, CA.QUANTITY, C.TOTAL_PRICE, P.DISCOUNT FROM PRODUCT P, CART_DETAILS CA, CART C WHERE C.CART_ID = CA.CART_ID AND C.CUST_ID = (SELECT CUST_ID FROM CUSTOMER C2 WHERE C2.EMAIL_ID = ('vielkawoodard@icloud.net')) AND P.PRODUCT_ID = CA.PRODUCT_ID;

| | PRODUCT_ID | PRODUCT_NAME | UNIT_PRICE | QUANTITY | TOTAL_PRICE | DISCOUNT |
|---|------------|---------------|------------|----------|-------------|----------|
| ▶ | P101 | cosmetics02 | 172 | 341 | 192160 | 0 |
| | P120 | stationery06 | 91 | 179 | 192160 | 0 |
| | P130 | electronics01 | 13 | 390 | 192160 | 0 |
| | P170 | toys11 | 133 | 279 | 192160 | 0 |
| | P249 | sports15 | 198 | 379 | 192160 | 0 |

- ❖ CREATE VIEW PRODUCT_VIEW AS SELECT PRODUCT_NAME, CATEGORY_ID FROM PRODUCT WHERE PRODUCT_QUANTITY < 20;
- ❖ DELETE FROM REVIEW R WHERE PRODUCT_ID = ('P129') AND R.CUST_ID = (SELECT CUST_ID FROM CUSTOMER C1 WHERE C1.EMAIL_ID = ('vielkawoodard@icloud.net'));
- ❖ SELECT DISTINCT PRODUCT_NAME, O.UNIT_PRICE, QUANTITY FROM PRODUCT P, ORDER_DETAILS O WHERE P.PRODUCT_ID = O.PRODUCT_ID AND O.ORDER_ID = 'O184';

| Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
|-------------|---------------|------------|--------------------|
| | PRODUCT_NAME | UNIT_PRICE | QUANTITY |
| ▶ | sports15 | 195 | 178 |
| | grocery11 | 38 | 117 |
| | pharmacy10 | 182 | 442 |
| | electronics10 | 174 | 353 |
| | grocery14 | 98 | 311 |
| | clothes05 | 197 | 287 |
| | toys05 | 83 | 465 |
| | toys13 | 24 | 306 |
| | clothes02 | 144 | 398 |

- ❖ SELECT distinct ORDER_ID, ORDER_TIME FROM PLACES_ORDER, CUSTOMER WHERE PLACES_ORDER.CUST_ID = (SELECT CUST_ID FROM CUSTOMER C1 WHERE C1.EMAIL_ID = ('vielkawoodard@icloud.net'));

| | ORDER_ID | ORDER_TIME |
|---|----------|------------|
| ▶ | O184 | 14:52:57 |
| | O140 | 18:43:35 |
| | O136 | 23:02:50 |

Embedded SQL Queries:

```
def proceed():
    sql2 = '''SELECT * FROM CART_DETAILS WHERE CART_ID = (%s)'''
    cur.execute(sql2,"CA" + cust_id[1:])
    f = cur.fetchall()
    sql3 = '''SELECT DISTINCT ORDER_ID FROM ORDERS'''
    cur.execute(sql3)
    f1 = cur.fetchall()
    order_id = "0" + str(int(f1[-1][0][1:]) + 1)
    for i in f:
        sql = '''INSERT INTO ORDER_DETAILS VALUES(%s,%s,%s,%s,%s)'''
        val = (order_id,f1[i][1],f1[i][2],f1[i][3],f1[i][4])
        cur.execute(sql,val)
        mydb.commit()
    sql1 = '''DELETE FROM CART_DETAILS WHERE CART_ID = (%s)'''
    cur.execute(sql1,"CA" + cust_id[1:])
    mydb.commit()
```

❖

```
def cart():
    user = "vielkawoodard@icloud.net"
    sql = "SELECT CUST_ID FROM CUSTOMER WHERE EMAIL_ID = (%s)"
    cur.execute(sql, user)
    f = cur.fetchall()
    cust_id = f[0][0]
    try:
        sql1 = '''
        SELECT DISTINCT CA.PRODUCT_ID,PRODUCT_NAME,CA.UNIT_PRICE,CA.QUANTITY,C.TOTAL_PRICE,P.DISCOUNT FROM
        PRODUCT P, CART_DETAILS CA, CART C
        WHERE C.CART_ID = CA.CART_ID AND C.CUST_ID = (SELECT CUST_ID FROM CUSTOMER C2 WHERE C2.EMAIL_ID = (%s))
        AND P.PRODUCT_ID = CA.PRODUCT_ID'''
        cur.execute(sql1)
    except:
        pass
```

❖

```
ent2.delete(id=10, last=100)
try:
    sql1 = '''
    SELECT PRODUCT_ID,PRODUCT_NAME,PRODUCT_QUANTITY,UNIT_PRICE,DISCOUNT,UNIT_WEIGHT,IN_STOCK, CATEGORY_NAME
    FROM CATEGORY C, PRODUCT P WHERE P.CATEGORY_ID = (%s) AND C.CATEGORY_ID = P.CATEGORY_ID
    ORDER BY PRODUCT_ID'''
    cur.execute(sql1,id)
```

❖

```

frames = {}
dict_customers = {}
dict_employee = {}
sql1 = '''SELECT EMAIL_ID, C_PASSWORD FROM CUSTOMER;'''
cur.execute(sql1)
rows = cur.fetchall()
for i in rows:
    dict_customers[i[0]] = i[1]
mydb.commit()
sql1 = '''SELECT EMAIL_ID, A_PASSWORD FROM EMPLOYEE;'''
cur.execute(sql1)
rows = cur.fetchall()
for i in rows:
    dict_employee[i[0]] = i[1]
mydb.commit()
print(dict_customers)
print(dict_employee)

```

❖

```

mydb.commit()
sql1 = '''CREATE TABLE CUSTOMER(
    CUST_ID CHAR(4) PRIMARY KEY,
    EMAIL_ID VARCHAR(50) NOT NULL ,
    FIRST_NAME VARCHAR(20) NOT NULL,
    LAST_NAME VARCHAR(20) NOT NULL,
    C_PASSWORD VARCHAR(30) NOT NULL,
    CHECK (EMAIL_ID LIKE '%@%.%'),
    CHECK (LENGTH(C_PASSWORD)>1 ),
    CHECK (CUST_ID LIKE 'C%'));'''
cur.execute(sql1)

sql2 = ''' CREATE TABLE SUPPLIER(
    SUPP_ID CHAR(4) PRIMARY KEY,
    EMAIL_ID VARCHAR(50) NOT NULL,
    COMPANY_NAME VARCHAR(50) NOT NULL,
    CHECK (SUPP_ID LIKE 'S%'),
    CHECK (EMAIL_ID LIKE '%@%.%'));'''

```

❖

```

for i in l:
    id = i[1]
    y+=1
    pid = i[2]
    quantity = int(i[4])
    price = int(i[3])
    discount = int(i[5])
    sql = 'INSERT INTO cart_details values(%s,%s,%s,%s,%s);'
    val = (id,pid,quantity,price,discount)
    print(val)
    cur.execute(sql, val)
    mydb.commit()

```

Data Population in Tables:

| | CATEGORY_ID | CATEGORY_NAME |
|---|-------------|---------------|
| ▶ | CT100 | COSMETICS |
| | CT101 | GROCERY |
| | CT102 | CLOTHES |
| | CT103 | SHOES |
| | CT104 | TOYS |
| | CT105 | FURNITURE |
| | CT106 | ELECTRONICS |
| | CT107 | SPORTS |
| | CT108 | PHARMACY |
| | CT109 | STATIONERY |
| • | NULL | NULL |

| | CUST_ID | EMAIL_ID | FIRST_NAME | LAST_NAME | C_PASSWORD |
|---|---------|----------------------------------|------------|------------|-------------|
| ▶ | C100 | blythe_case7564@aol.net | Blythe | Case | NJJ74DHN7XA |
| | C101 | c.erickson6969@outlook.edu | Chelsea | Erickson | IYL06VQE3DK |
| | C102 | hebertcharity5210@yahoo.in | Charity | Hebert | BAY99IMX4JE |
| | C103 | martin.sean2757@google.in | Sean | Martin | GQP87JCN4EE |
| | C104 | a_powers@hotmail.co.in | Andrew | Powers | QQS72DET6LH |
| | C105 | ahumphrey5348@google.co.in | Aidan | Humphrey | ZPP02FZQ7TV |
| | C106 | mamber@yahoo.in | Amber | Mccullough | BYO19JFG6CI |
| | C107 | stricklandjeremy@outlook.org | Jeremy | Strickland | VCF76WIE6KX |
| | C108 | brady.vaughan8212@outlook.in | Vaughan | Brady | VIL98QEH3RX |
| | C109 | mreese@protonmail.net | Minerva | Reese | DMH10VKO8FQ |
| | C110 | joyce.cedric9316@protonmail.c... | Cedric | Joyce | YXZ44OYF7LY |
| | C111 | vielkawoodard@icloud.net | Vielka | Woodard | YQV56KKA5VF |
| | C112 | simpson.armand@hotmail.org | Armand | Simpson | FLW83BGR0MR |
| | C113 | wcurran@google.edu | Curran | Ware | SKQ85XS3JZ |
| | C114 | s-carev7228@ooodle.net | Solomon | Carev | VAI53DGP8SE |

| | SUPP_ID | EMAIL_ID | COMPANY_NAME |
|---|---------|--|-----------------------------|
| ▶ | S100 | vivamus.nibh@google.org | Ut Ipsum LLP |
| | S101 | amet@yahoo.org | Euismod Urna Nullam Limited |
| | S102 | mauris@yahoo.couk | Cras Dolor Company |
| | S103 | faucibus@protonmail.couk | Ut Eros Associates |
| | S104 | ut.odio@yahoo.org | Nunc Corporation |
| | S105 | donec.fringilla.donec@yahoo.edu | Gravida LLP |
| | S106 | sollicitudin.commodo.ipsum@outlook.com | Ultrices Inc. |
| | S107 | tempus.risus@outlook.ca | Luctus Incorporated |
| | S108 | tempor.augue@protonmail.couk | Dui Fusce Diam Foundation |
| | S109 | cras.dictum@yahoo.net | Lorem Foundation |
| | S110 | et.euismod.et@yahoo.org | Risus In Inc. |
| | S111 | aliquet.diam@yahoo.edu | Et Magnis Foundation |
| | S112 | id.sapien.cras@icloud.edu | Erat Consulting |
| | S113 | malesuada.malesuada@protonmail.ca | Nisi PC |
| | S114 | urna.nec.luctus@outlook.ca | Mauris Eu Ltd |

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

| | PERSONAL_ID | LOCALITY | CITY | STATE | COUNTRY | POSTAL_CODE | PHONE_NUMBER |
|---|-------------|---------------------------|---------------|----------------------|---------------|-------------|--------------|
| ▶ | S100 | 748 Algoma Crossing | Washington | District of Columbia | United States | 20210 | 2025627387 |
| | S101 | 59978 Blaine Parkway | Santa Fe | New Mexico | United States | 87592 | 5051838969 |
| | S102 | 44 Twin Pines Pass | Lubbock | Texas | United States | 79405 | 8063798402 |
| | S103 | 811 Barnett Way | Plano | Texas | United States | 75074 | 4696165874 |
| | S104 | 6 New Castle Junction | Bradenton | Florida | United States | 34282 | 9415390592 |
| | S105 | 3386 Susan Court | Fresno | California | United States | 93762 | 5595445302 |
| | S106 | 8679 Bultman Park | Washington | District of Columbia | United States | 20337 | 2027748968 |
| | S107 | 7338 Scoville Trail | Philadelphia | Pennsylvania | United States | 19146 | 2672785872 |
| | S108 | 05 Dawn Crossing | Laredo | Texas | United States | 78044 | 9562685217 |
| | S109 | 84530 Brickson Park Point | Atlanta | Georgia | United States | 31136 | 4049011429 |
| | S110 | 35609 Briar Crest Pass | Albuquerque | New Mexico | United States | 87121 | 5057410368 |
| | S111 | 41751 Kim Plaza | Fort Worth | Texas | United States | 76147 | 8175194021 |
| | S112 | 51 Barby Drive | Colorado S... | Colorado | United States | 80940 | 7194463445 |
| | S113 | 8769 8th Point | Myrtle Beach | South Carolina | United States | 29579 | 8431585241 |
| | S114 | 86802 Packers Circle | Charlotte | North Carolina | United States | 28247 | 7046193715 |

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

| | PERSONAL_ID | LOCALITY | CITY | STATE | COUNTRY | POSTAL_CODE | PHONE_NUMBER |
|---|-------------|--------------------------|-----------------|----------------|---------------|-------------|--------------|
| ▶ | C100 | 66789 Emmet Way | Anaheim | California | United States | 92812 | 7147682665 |
| | C101 | 208 Arizona Drive | Peoria | Illinois | United States | 61656 | 3094154507 |
| | C102 | 7104 Kings Circle | Minneapolis | Minnesota | United States | 55470 | 6123244810 |
| | C103 | 9216 Monica Alley | Tampa | Florida | United States | 33633 | 8138731934 |
| | C104 | 352 Melby Terrace | Raleigh | North Carolina | United States | 27635 | 9194272589 |
| | C105 | 1 Drewry Hill | Pueblo | Colorado | United States | 81015 | 7193166493 |
| | C106 | 07 Kingsford Circle | Portland | Maine | United States | 04109 | 2074025007 |
| | C107 | 49 Burning Wood Circle | Los Angeles | California | United States | 90010 | 4081257209 |
| | C108 | 80028 Vermont Circle | Denver | Colorado | United States | 80299 | 7205900846 |
| | C109 | 77 Calypso Crossing | Fort Lauderdale | Florida | United States | 33355 | 7544123991 |
| | C110 | 2606 Springs Avenue | Las Cruces | New Mexico | United States | 88006 | 5051292719 |
| | C111 | 07493 Blaine Plaza | Oklahoma City | Oklahoma | United States | 73173 | 4053508713 |
| | C112 | 04 Rockefeller Lane | Pittsburgh | Pennsylvania | United States | 15250 | 4127622072 |
| | C113 | 69962 Clyde Gallagher... | Fresno | California | United States | 93726 | 5597597685 |
| | C114 | 53378 Forster Hill | Spokane | Washington | United States | 99220 | 5096350265 |

| | PRODUCT_ID | PRODUCT_NAME | PRODUCT_QUANTITY | UNIT_PRICE | UNIT_WEIGHT | DISCOUNT | CATEGORY_ID | IN_STOCK |
|---|------------|--------------|------------------|------------|-------------|----------|-------------|----------|
| ▶ | p100 | cosmetics01 | 34 | 357 | 126 | 0 | CT100 | 1 |
| | p101 | cosmetics02 | 165 | 189 | 250 | 0 | CT100 | 1 |
| | p102 | cosmetics03 | 74 | 186 | 1199 | 0 | CT100 | 1 |
| | p103 | cosmetics04 | 169 | 465 | 482 | 0 | CT100 | 1 |
| | p104 | cosmetics05 | 189 | 425 | 862 | 0 | CT100 | 1 |
| | p105 | cosmetics06 | 179 | 363 | 1274 | 0 | CT100 | 1 |
| | p106 | cosmetics07 | 103 | 235 | 351 | 0 | CT100 | 1 |
| | p107 | cosmetics08 | 153 | 371 | 633 | 0 | CT100 | 1 |
| | p108 | cosmetics09 | 104 | 314 | 630 | 0 | CT100 | 1 |
| | p109 | cosmetics10 | 176 | 41 | 651 | 0 | CT100 | 1 |
| | p110 | cosmetics11 | 80 | 171 | 1018 | 0 | CT100 | 1 |
| | p111 | cosmetics12 | 14 | 244 | 1332 | 0 | CT100 | 1 |
| | p112 | cosmetics13 | 12 | 42 | 422 | 0 | CT100 | 1 |
| | p113 | cosmetics14 | 97 | 479 | 223 | 0 | CT100 | 1 |
| | p114 | cosmetics15 | 58 | 114 | 447 | 0 | CT100 | 1 |

| SHIP_ID | PHONE_NUMBER | COMPANY_NAME |
|---------|--------------|-------------------------------|
| SH100 | 7043387992 | Hauck Inc |
| SH101 | 5387327325 | Dickens-Roob |
| SH102 | 3145160622 | Smith, Kunze and Bartell |
| SH103 | 7162458779 | Purdy and Sons |
| SH104 | 5638290989 | Maggio, Lemke and Gulowski |
| SH105 | 7322760954 | Nicolas-Mertz |
| SH106 | 6934377874 | Zulauf-Fahey |
| SH107 | 7224388870 | Marquardt-Collins |
| SH108 | 2005185339 | Wunsch-Toy |
| SH109 | 2072617247 | White, Gislason and Torphy |
| SH110 | 6816182398 | Weber LLC |
| SH111 | 9816858382 | Pagac Inc |
| SH112 | 9118463274 | Homenick-Casper |
| SH113 | 3556213408 | Harris-Quitzon |
| SH114 | 5461319656 | Reichel, Douglas and Reynolds |

| ORDER_ID | BILL_ID | SHIPPING_ID | CONSIGNMENT_NUMBER | SHIPPED_DATE | ORDER_DATE |
|----------|---------|-------------|--------------------|---------------------|---------------------|
| O100 | B100 | SH100 | CW100 | 2021-05-21 06:16:45 | 2021-05-26 06:16:45 |
| O101 | B101 | SH101 | CW101 | 2021-06-18 03:25:48 | 2021-06-23 03:25:48 |
| O102 | B102 | SH102 | CW102 | 2021-05-29 13:08:55 | 2021-06-03 13:08:55 |
| O103 | B103 | SH103 | CW103 | 2021-08-09 13:48:03 | 2021-08-14 13:48:03 |
| O104 | B104 | SH104 | CW104 | 2021-01-10 09:40:10 | 2021-01-15 09:40:10 |
| O105 | B105 | SH105 | CW105 | 2021-11-10 03:58:12 | 2021-11-15 03:58:12 |
| O106 | B106 | SH106 | CW106 | 2021-11-14 23:26:29 | 2021-11-19 23:26:29 |
| O107 | B107 | SH107 | CW107 | 2021-11-15 10:01:05 | 2021-11-20 10:01:05 |
| O108 | B108 | SH108 | CW108 | 2021-10-18 07:37:55 | 2021-10-23 07:37:55 |
| O109 | B109 | SH109 | CW109 | 2021-04-21 15:20:11 | 2021-04-26 15:20:11 |
| O110 | B110 | SH110 | CW110 | 2021-09-03 09:40:24 | 2021-09-08 09:40:24 |
| O111 | B111 | SH111 | CW111 | 2021-07-04 00:09:43 | 2021-07-09 00:09:43 |
| O112 | B112 | SH112 | CW112 | 2021-10-03 07:08:29 | 2021-10-08 07:08:29 |
| O113 | B113 | SH113 | CW113 | 2021-10-14 02:52:50 | 2021-10-19 02:52:50 |
| O114 | B114 | SH114 | CW114 | 2021-07-09 05:43:52 | 2021-07-14 05:43:52 |

| PAYMENT_ID | ORDER_ID | CUST_ID | PAYMENT_METHOD | AMOUNT_PAID |
|------------|----------|---------|----------------|-------------|
| PA100 | O100 | C100 | credit card | 912 |
| PA101 | O101 | C101 | credit card | 456 |
| PA102 | O102 | C102 | emi | 1873 |
| PA103 | O103 | C103 | credit card | 2182 |
| PA104 | O104 | C104 | netbanking | 2218 |
| PA105 | O105 | C105 | netbanking | 2029 |
| PA106 | O106 | C106 | credit card | 783 |
| PA107 | O107 | C107 | netbanking | 2207 |
| PA108 | O108 | C108 | paytm | 717 |
| PA109 | O109 | C109 | paytm | 331 |
| PA110 | O110 | C110 | credit card | 1504 |
| PA111 | O111 | C111 | netbanking | 1246 |
| PA112 | O112 | C112 | debit card | 925 |
| PA113 | O113 | C113 | paytm | 598 |
| PA114 | O114 | C114 | netbanking | 896 |

Referential integrity constraints:

| Table Name | Attributes | Data types | Constraints | Keys | | | | | |
|----------------------|------------------|-------------|-------------|--|--|--|--|--|--|
| CUSTOMER | CUST_ID | CHAR(4) | NOT NULL | PRIMARY KEY | | | | | |
| | FIRST NAME | VARCHAR(20) | NOT NULL | | | | | | |
| | LAST NAME | VARCHAR(20) | NOT NULL | | | | | | |
| | EMAIL_ID | VARCHAR(30) | NOT NULL | | | | | | |
| | C_PASSWORD | VARCHAR(30) | NOT NULL | | | | | | |
| CUSTOMER_INFORMATION | PERSONAL_ID | CHAR(4) | NOT NULL | FOREIGN_KEY REFERENCES CUSTOMER(CUST_ID) ON DELETE CASCADE | | | | | |
| | PHONE_NUMBER | CHAR(10) | NOT NULL | | | | | | |
| | LOCALITY | VARCHAR(50) | NOT NULL | | | | | | |
| | CITY | VARCHAR(20) | NOT NULL | | | | | | |
| | STATE | VARCHAR(20) | NOT NULL | | | | | | |
| SUPPLIER_INFORMATION | COUNTRY | VARCHAR(20) | NOT NULL | | | | | | |
| | POSTAL_CODE | CHAR(5) | NOT NULL | | | | | | |
| | PERSONAL_ID | CHAR(4) | NOT NULL | FOREIGN_KEY REFERENCES CUSTOMER(CUST_ID) ON DELETE CASCADE | | | | | |
| | PHONE_NUMBER | CHAR(10) | NOT NULL | | | | | | |
| | LOCALITY | VARCHAR(50) | NOT NULL | | | | | | |
| SUPPLIER | CITY | VARCHAR(20) | NOT NULL | | | | | | |
| | STATE | VARCHAR(20) | NOT NULL | | | | | | |
| | COUNTRY | VARCHAR(20) | NOT NULL | | | | | | |
| | POSTAL_CODE | CHAR(5) | NOT NULL | | | | | | |
| | | | | | | | | | |
| PRODUCT | SUPP_ID | CHAR(4) | NOT NULL | PRIMARY KEY | | | | | |
| | COMPANY_NAME | VARCHAR(30) | NOT NULL | | | | | | |
| | EMAIL_ID | VARCHAR(30) | NOT NULL | | | | | | |
| | PRODUCT_ID | CHAR(4) | NOT NULL | PRIMARY KEY | | | | | |
| | NAME | VARCHAR(20) | NOT NULL | | | | | | |
| EMPLOYEE | PRODUCT_QUANTITY | INT | NOT NULL | | | | | | |
| | UNIT PRICE | INT | NOT NULL | | | | | | |
| | UNIT WEIGHT | INT | NOT NULL | | | | | | |
| | IN STOCK | BOOLEAN | NOT NULL | | | | | | |
| | ADMIN_ID | CHAR(4) | NOT NULL | PRIMARY KEY | | | | | |
| | FIRST_NAME | VARCHAR(20) | NOT NULL | | | | | | |
| | LAST_NAME | VARCHAR(20) | NOT NULL | | | | | | |
| | ADMIN_ROLES | BOOLEAN | NOT NULL | | | | | | |
| | EMAIL_ID | VARCHAR(30) | NOT NULL | | | | | | |
| | A_PASSWORD | VARCHAR(30) | NOT NULL | | | | | | |

| | | | | | | | | | |
|------|----------------|---------|----------|--|--|--|--|--|--|
| CART | CART_ID | CHAR(5) | NOT NULL | PRIMARY KEY | | | | | |
| | TOTAL_PRICE | INTEGER | NOT NULL | | | | | | |
| | TOTAL_QUANTITY | INTEGER | NOT NULL | | | | | | |
| | ORDER_ID | CHAR(4) | NOT NULL | PRIMARY KEY | | | | | |
| | CUST_ID | CHAR(4) | NOT NULL | FOREIGN KEY (CUST_ID) REFERENCES CUSTOMER(CUST_ID) ON DELETE CASCADE | | | | | |
| | SUPP_ID | CHAR(4) | NOT NULL | FOREIGN KEY (SUPP_ID) REFERENCES SUPPLIER(SUPP_ID) ON DELETE CASCADE | | | | | |

| | | | | | | | | | |
|---------------|--------------------|-------------|----------|---|--|--|--|--|--|
| ORDERS | BILL_ID | CHAR(4) | NOT NULL | FOREIGN KEY (BILL_ID) REFERENCES BILLS(BILL_ID) ON DELETE CASCADE | | | | | |
| | SHIPPING_ID | CHAR(5) | NOT NULL | FOREIGN KEY (SHIPPING_ID) REFERENCES SHIPPER(SHIP_ID) ON DELETE CASCADE | | | | | |
| | ORDER_DATE | DATE | NOT NULL | | | | | | |
| | SHIPPED_DATE | DATE | | | | | | | |
| | CONSIGNMENT_NUMBER | INT | | | | | | | |
| CATEGORY | CATEGORY_ID | CHAR(5) | NOT NULL | PRIMARY KEY | | | | | |
| | CATEGORY_NAME | VARCHAR(30) | NOT NULL | | | | | | |
| BILLS | BILL_ID | CHAR(4) | NOT NULL | PRIMARY KEY | | | | | |
| | BILL_DATE | DATE | NOT NULL | | | | | | |
| | CUST_ID | CHAR(4) | NOT NULL | FOREIGN KEY (CUST_ID) REFERENCES CUSTOMER(CUST_ID); | | | | | |
| | COUPON_ID | CHAR(5) | NOT NULL | FOREIGN KEY (COUPON_ID) REFERENCES COUPONS(COUPON_ID); | | | | | |
| | PAYMENT_ID | CHAR(4) | NOT NULL | FOREIGN KEY (PAYMENT_ID) REFERENCES PAYMENT(PAYMENT_ID); | | | | | |
| COUPONS | COUPON_ID | CHAR(5) | NOT NULL | PRIMARY KEY | | | | | |
| | PAYMENT_METHOD | VARCHAR(20) | NOT NULL | | | | | | |
| | DISCOUNT | INTEGER | NOT NULL | | | | | | |
| | ORDER_ID | CHAR(4) | NOT NULL | FOREIGN KEY (ORDER_ID) REFERENCES ORDERS(ORDER_ID); | | | | | |
| | PRODUCT_ID | CHAR(4) | NOT NULL | FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT(PRODUCT_ID); | | | | | |
| ORDER_DETAILS | UNIT_PRICE | INTEGER | NOT NULL | | | | | | |
| | QUANTITY | INTEGER | NOT NULL | | | | | | |
| | DISCOUNT | INTEGER | NOT NULL | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| REVIEW | CUST_ID | CHAR(4) | NOT NULL | FOREIGN KEY (CUST_ID) REFERENCES CUSTOMER(CUST_ID) ON DELETE CASCADE | | | | | |
| | PRODUCT_ID | CHAR(4) | NOT NULL | FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT(PRODUCT_ID) ON DELETE CASCADE | | | | | |
| | RATING | INTEGER | NOT NULL | | | | | | |
| PAYMENT | PAYMENT_ID | CHAR(4) | NOT NULL | PRIMARY KEY | | | | | |
| | ORDER_ID | CHAR(4) | NOT NULL | FOREIGN KEY (ORDER_ID) REFERENCES ORDERS(ORDER_ID) ON DELETE CASCADE | | | | | |
| | CUST_ID | CHAR(4) | NOT NULL | FOREIGN KEY (CUST_ID) REFERENCES CUSTOMER(CUST_ID) ON DELETE CASCADE | | | | | |
| | PAYMENT_METHOD | VARCHAR(20) | NOT NULL | | | | | | |
| | AMOUNT_PAID | INTEGER | NOT NULL | | | | | | |

| | | | | | | | | |
|--------------|--------------|-------------|-----------|---|--|--|--|--|
| SUPPLIES | SUPP_ID | CHAR(4) | NOT NULL | FOREIGN KEY (SUPP_ID) REFERENCES SUPPLIER(SUPP_ID) ON DELETE CASCADE | | | | |
| | PRODUCT_ID | CHAR(4) | NOT NULL | FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT(PRODUCT_ID) ON DELETE CASCADE | | | | |
| APPLIED_COUP | COUPON_ID | CHAR(4) | NOT NULL | FOREIGN KEY (COUPON_ID) REFERENCES COUPONS(COUPON_ID) ON DELETE CASCADE | | | | |
| | CART_ID | CHAR(4) | NOT NULL | FOREIGN KEY (CART_ID) REFERENCES CART(CART_ID) ON DELETE CASCADE | | | | |
| PLACES_ORDER | ORDER_ID | CHAR(4) | NOT NULL | FOREIGN KEY (ORDER_ID) REFERENCES ORDERS(ORDER_ID) ON DELETE CASCADE | | | | |
| | CUST_ID | CHAR(4) | NOT NULL | FOREIGN KEY (CUST_ID) REFERENCES CUSTOMER(CUST_ID) ON DELETE CASCADE | | | | |
| SHIPPER | ORDER_TIME | TIME | NOT NULL, | | | | | |
| | SHIP_ID | CHAR(4) | NOT NULL | PRIMARY KEY | | | | |
| | PHONE_NUMBER | CHAR(10) | NOT NULL, | | | | | |
| CART_DETAILS | COMPANY_NAME | VARCHAR(20) | NOT NULL, | | | | | |
| | CART_ID | CHAR(4) | NOT NULL | FOREIGN KEY (CART_ID) REFERENCES ORDERS(ORDER_ID), | | | | |
| | PRODUCT_ID | CHAR(4) | NOT NULL | FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT(PRODUCT_ID); | | | | |
| | UNIT_PRICE | INTEGER | NOT NULL | | | | | |
| | QUANTITY | INTEGER | NOT NULL | | | | | |

| | | | | | | | | |
|----------|---------|----------|--|--|--|--|--|--|
| DISCOUNT | INTEGER | NOT NULL | | | | | | |
|----------|---------|----------|--|--|--|--|--|--|

Views, Grants and Triggers:

❖ Views

```

try:
    sql1 = '''CREATE VIEW SUPPLIER_DETAILS AS
              SELECT SUPP_ID,COMPANY_NAME,EMAIL_ID,LOCALITY,CITY,STATE,COUNTRY,POSTAL_CODE,PHONE_NUMBER
              FROM SUPPLIER,SUPPLIER_INFORMATION WHERE SUPPLIER.SUPP_ID = SUPPLIER_INFORMATION.PERSONAL_ID;'''
    cur.execute(sql1)
except:
    pass

```

```

try:
    sql1 = '''CREATE VIEW SHIPPER_DETAILS AS
              SELECT SHIP_ID,COMPANY_NAME,PHONE_NUMBER
              FROM SHIPPER;'''
    cur.execute(sql1)
except:
    pass

```

```

try:
    sql1 = '''CREATE VIEW PRODUCT_1 AS
              SELECT distinct P.PRODUCT_ID,PRODUCT_NAME,PRODUCT_QUANTITY,UNIT_PRICE,UNIT_WEIGHT,
              S.COMPANY_NAME, C.CATEGORY_NAME
              FROM REVIEW AS R, SUPPLIER AS S, PRODUCT AS P, CATEGORY AS C, SUPPLIES AS Q
              WHERE P.Product_ID = Q.product_ID and s.supp_id = q.supp_id AND C.CATEGORY_ID = P.CATEGORY_ID
              order by p.product_id;'''
    cur.execute(sql1)

    sql2 = '''create view product_2 as
              (Select p1.product_id, avg(rating) as reviews from review r1, product p1 where r1.product_id = p1.product_id
              group by p1.product_id);'''
    cur.execute(sql2)

    sql3 = '''create view product_details as
              select product_1.PRODUCT_ID,PRODUCT_NAME,PRODUCT_QUANTITY,UNIT_PRICE,UNIT_WEIGHT,
              COMPANY_NAME, CATEGORY_NAME, reviews from product_1,product_2
              where product_1.product_id = product_2.product_id
              order by product_id;'''

```

❖ Grants

```

CREATE USER IF NOT EXISTS 'customer'@'localhost'
IDENTIFIED BY 'customer';
GRANT INSERT
ON retailstore.cart
TO 'customer'@'localhost';

```

```
SHOW GRANTS FOR 'customer'@'localhost';  
REVOKE INSERT  
ON retailstore.employee  
FROM 'ars'@'localhost';  
|
```

```
CREATE USER IF NOT EXISTS 'customer'@'localhost'  
IDENTIFIED BY 'admin';  
GRANT SELECT  
ON customer  
TO 'customer'@'localhost'; |
```

```
CREATE USER IF NOT EXISTS 'customer'@'localhost'  
IDENTIFIED BY 'admin';  
GRANT SELECT  
on customer_information |  
TO 'customer'@'localhost';
```

```
CREATE USER IF NOT EXISTS 'customer'@'localhost'  
IDENTIFIED BY 'admin';  
GRANT SELECT  
on cart  
TO 'customer'@'localhost'; |
```

```
CREATE USER IF NOT EXISTS 'customer'@'localhost'  
IDENTIFIED BY 'admin';  
GRANT SELECT  
on cart_details |  
TO 'customer'@'localhost';
```

```
CREATE USER IF NOT EXISTS 'employee'@'localhost'  
IDENTIFIED BY 'admin';  
GRANT insert  
on product  
TO 'employee'@'localhost'; |
```

```
grant select on orders TO 'customer'@'localhost';  
grant select on order_details TO 'customer'@'localhost';  
grant select on review TO 'customer'@'localhost'; |
```

```
grant select on customer TO 'employee'@'localhost';
grant select on customer_information TO 'employee'@'localhost';
grant select on cart TO 'employee'@'localhost';
grant select on cart_details TO 'employee'@'localhost';
grant select on orders TO 'employee'@'localhost';
grant select on order_details TO 'employee'@'localhost';
grant select on product TO 'employee'@'localhost';
```

```
grant insert on customer TO 'employee'@'localhost';
grant insert on customer_information TO 'employee'@'localhost';
grant insert on cart TO 'employee'@'localhost';
grant insert on cart_details TO 'employee'@'localhost';
grant insert on orders TO 'employee'@'localhost';
grant insert on order_details TO 'employee'@'localhost';
grant insert on product TO 'employee'@'localhost';
```

```
grant insert on supplier TO 'employee'@'localhost';
grant insert on supplier_information TO 'employee'@'localhost';
grant insert on shipper TO 'employee'@'localhost';
grant insert on applied_coup TO 'employee'@'localhost';
grant insert on places_order TO 'employee'@'localhost';
grant insert on adds_product TO 'employee'@'localhost';
grant insert on category TO 'employee'@'localhost';
```

❖ Triggers

```
DROP TRIGGER IF EXISTS `retailstore`.`cart_upd`;
DELIMITER $$
USE `retailstore`$$
CREATE DEFINER = CURRENT_USER TRIGGER
`online_banking_system`.`cart_upd` BEFORE INSERT
ON `cart_details` FOR EACH ROW
BEGIN
IF NEW.quantity >= NEW.product_quantity THEN
SET NEW.quantity = NEW.product_quantity;
END IF;
END$$
DELIMITER ;
```

```
BEGIN
```

```
IF NEW.total_price > 10000 THEN  
    SET NEW.total_price = NEW.total_price *0.9  
ELSEIF NEW.total_price > 20000 THEN  
    SET NEW.total_price = NEW.total_price *0.85  
else SET NEW.total_price = NEW.total_price *0.8  
    end if
```

```
END;
```

```
BEGIN
```

```
IF NEW.quantity > 100 THEN  
    SET NEW.total_price = NEW.total_price *0.9  
ELSEIF NEW.quantity < 10 THEN  
    SET NEW.total_price = NEW.total_price + 100  
    end if
```

```
END;
```

```
BEGIN
```

```
declare coupon_discount = 0.1  
IF NEW.coupon_applied==1 THEN  
    SET NEW.total_price = NEW.total_price*coupon_discount  
ELSE  
    SET NEW.total_price = NEW.total_price  
    end if
```

```
END;
```

Relational Algebra:

① Customer ID's of all customers with last name "Hero".

$\pi_{\text{cust-id}} (\sigma_{\text{last-name} = \text{"Hero"}} (\text{customer}))$

② All the products with unit price greater than 500 and unit weight less than 1.5 Kg.

$\pi_{\text{product-id, name, in stock}} (\sigma_{\text{unit price} > 500 \wedge \text{unit weight} < 1.5} (\text{product}))$

③ Delete the record of ~~supplier~~ Supplier ID = "S103"

$\text{Supplier} \leftarrow \text{Supplier} - \sigma_{\text{supplier-id} = \text{"S103"}} (\text{Supplier})$

③ Delete the record of ~~supplier~~ Supplier ID = "S103"

$\text{Supplier} \leftarrow \text{Supplier} - \sigma_{\text{supplier-id} = \text{"S103"}} (\text{Supplier})$

④ Insert a new product with details:-

ID = "P109"

Name - Product X

Quantity - 106

Unit price - 500

Unit weight - 200

$\text{Product} \leftarrow \text{Product} \cup \{(\text{"P109"}, \text{"Product X"}, 106, 500, 200)\}$

DATE

⑥ Find the most expensive product in the product list-

$\pi_{\text{unit price}} (\text{Products}) - \pi_{\text{product-id, name, in stock}} (\sigma_{\text{products-unit price} < \text{d-price}} (\text{products} \times \rho_d (\text{products})))$

⑦ List of all the customers ~~differs~~ with postal code = "110020"

$$\pi_{\text{Cust-id, Firstname, Last name}} (\sigma_{\text{Postal code} = "110020"} (\text{Customers} \bowtie \text{Customer info}))$$

⑧ List of all the products that are in stock with price greater than 1500.

$$\pi_{\text{product-id, Name, Quantity}} (\sigma_{\text{Instock} = "True"} (\text{Product})) \cap$$

$$\pi_{\text{product-id, Name, Quantity}} (\sigma_{\text{Unit price} > 1500} (\text{Product}))$$

⑨ List of all the products ^{ordered by} ~~in the sort of~~ all the customers.

$$\pi_{\text{Customer-Firstname, Customer-Last name, Customer-cust-id}} (\sigma_{\text{Customers-cust-id} = \text{Orders-cust-id}} (\text{Orders} \times \text{Customer}))$$

~~product-id, Name~~ product-name.

⑩ Products belonging to category id = "CT102" or products ~~costing~~ costing more than 1000.

$$\pi_{\text{product-id, Name, instock}} (\sigma_{\text{category-id} = "CT102"} (\text{Product})) \cup$$

$$\pi_{\text{product-id, Name, instock}} (\sigma_{\text{Unit price} > 1000} (\text{Product}))$$

CLASSMATE PAGE

Front End Screens:

3121

To view the full application,click the button below!

Click Me!

Welcome to Amazon :)

Click Me! f1

amazon

CUSTOMER LOGIN

EMPLOYEE LOGIN



For new customers, sign up below!

SIGN UP!

Welcome to Amazon :)

| | |
|---------------|--------------------------|
| First Name | <input type="text"/> |
| Last Name | <input type="text"/> |
| Email Address | <input type="text"/> |
| Password | <input type="password"/> |

Next

Back

Welcome to Amazon :)

| | |
|---------------|--------------------------|
| Email Address | <input type="text"/> |
| Password | <input type="password"/> |

| | |
|--------|------|
| Submit | Back |
|--------|------|

Welcome to Amazon :)

Hello, John

View Products by Category

View Previous Orders

Browse and Shop!

Update Personal Information

Delete Reviews

Add Review

Back

Welcome to Amazon :)

Hello, John

Add Products

Delete Products

Update Products

View Stakeholders

View Sales

Update Entities

Place Order for Supplies

Delete Entities

Logout

Welcome to Amazon :)

Hello, John

Product Name

Unit Weight

Unit Price

Submit

Back

Cosmetics

Grocery

Clothes

Shoes

Toys

Furniture

Electronics

Sports

Pharmacy

Stationery

Welcome to Amazon :)

Hello, John

Delete by Product ID

Delete by Product Name

Delete by Category

Back

Welcome to Amazon :)

Hello, John

Enter Product ID

Submit

Back

Welcome to Amazon :)

Hello, John

View Customers

View Suppliers

View Shippers

View Products

Back

Welcome to Amazon :)

Hello, John

| Custom | First Name | Last Name | Email Address | Locality | City | State | Country | Postal Code | Contact |
|--------|------------|------------|-------------------------|------------------------|-----------------|----------------|---------------|-------------|------------|
| C100 | Blythe | Case | blythe_case7564@aol.net | 66789 Emmet Way | Anaheim | California | United States | 92812 | 7147682665 |
| C101 | Chelsea | Erickson | c.erickson6969@outlook. | 208 Arizona Drive | Peoria | Illinois | United States | 61656 | 3094154507 |
| C102 | Charity | Hebert | hebertcharity5210@yaho | 7104 Kings Circle | Minneapolis | Minnesota | United States | 55470 | 6123244810 |
| C103 | Sean | Martin | martin.sean2757@google | 9216 Monica Alley | Tampa | Florida | United States | 33633 | 8138731934 |
| C104 | Andrew | Powers | a_powers@hotmail.co.in | 352 Melby Terrace | Raleigh | North Carolina | United States | 27635 | 9194272589 |
| C105 | Aidan | Humphrey | ahumphrey5348@google | 1 Drewry Hill | Pueblo | Colorado | United States | 81015 | 7193166493 |
| C106 | Amber | Mccullough | mamber@yahoo.in | 07 Kingsford Circle | Portland | Maine | United States | 04109 | 2074025007 |
| C107 | Jeremy | Strickland | stricklandjeremy@outloo | 49 Burning Wood Circle | Los Angeles | California | United States | 90010 | 4081257209 |
| C108 | Vaughan | Brady | brady.vaughan8212@out | 80028 Vermont Circle | Denver | Colorado | United States | 80299 | 7205900846 |
| C109 | Minerva | Reese | mreese@protonmail.net | 77 Calypso Crossing | Fort Lauderdale | Florida | United States | 33355 | 7544123991 |

Back

Welcome to Amazon :)

Hello, John

| Product ID | Name | Quantity | Price | Weight | Company | Category | Avg Rating |
|------------|-------------|----------|-------|--------|--------------------------|-----------|------------|
| p100 | cosmetics01 | 34 | 357 | 126 | Placerat Eget Associates | COSMETICS | 2.5000 |
| p101 | cosmetics02 | 165 | 189 | 250 | Tristique Neque Inc. | COSMETICS | 2.8571 |
| p102 | cosmetics03 | 74 | 186 | 1199 | Suspendisse Eleifend Cra | COSMETICS | 2.5000 |
| p103 | cosmetics04 | 169 | 465 | 482 | Vel Sapien Imperdiet LLC | COSMETICS | 3.6000 |
| p104 | cosmetics05 | 189 | 425 | 862 | Ultrices Inc. | COSMETICS | 2.0000 |
| p105 | cosmetics06 | 179 | 363 | 1274 | Erat Consulting | COSMETICS | 3.0000 |
| p106 | cosmetics07 | 103 | 235 | 351 | Integer Vitae PC | COSMETICS | 2.7500 |
| p107 | cosmetics08 | 153 | 371 | 633 | Donec Corp. | COSMETICS | 3.0000 |
| p108 | cosmetics09 | 104 | 314 | 630 | Integer Vitae PC | COSMETICS | 3.5714 |
| p109 | cosmetics10 | 176 | 41 | 651 | Dignissim Tempor Consu | COSMETICS | 3.6000 |

Back

Work Distribution:

Since, it was impossible to mention all the csv files and the data population as well as the frontend code, we have provided the Github repository link for the same. It has been made private to avoid plagiarism.

Link to repository:

<https://github.com/jayanpahuja20/RetailStore>

Ayush Sharma - Scope of Project, Stakeholders, ER Diagram, Entities Relationships, Relation Schema, View and Grants (Minor Work), Query Optimisation, Indexing, Triggers, Relational Algebra, Data Population, Documentation,

Jayan Pahuja - ER Diagram, Entities and relationships, Views and grants (Major Work), Sql queries, Embedded SQL queries, Indexing, Triggers, Data population, Documentation, Frontend (Major Work)

Aaryendra - ER Diagram (Major Work), Entities and relationships, Views and grants (Minor Work), Sql queries, Triggers, Data Population, Documentation

Aayush Gakhar - ER Diagram, Entities and relationships, Relational schema, Views and grants (Minor Work), Sql queries, Embedded sql queries, Indexing, Triggers, Relational algebra, Frontend