Table of Contents

I. Storyline

1.We have created a small Amazon Database Management System, which represents the database management system of the actual Amazon's E-commerce Website.

2.Amazon works in a Business to Consumer (B2C) mode and we have covered all the basics work modes under such a system.

3. The Amazon database keeps a track of the Product and stores their information in the database.

3.1 The name, product id, brand, cost, availability & discount is stored in the database.

3.2 Each product has a unique id.

3.3 A product gets added to the cart when an order is placed by a Customer.

4. The Amazon Database needs to keep a record of its Customers and the Orders placed by every Customer:

4.1 Every customer has an ID, name, email address, phone number and address.

4.2 The database keeps track of orders placed by the customer.

4.3 Every Order has a unique Order ID

4.4 It keeps track of the order date as well as the delivery date

5. The database must keep track of the Suppliers:

5.1 Each Supplier has an ID and a name.

5.2 Every Supplier has a unique Supplier ID.

5.3 A Supplier supplies one or more products of one or many brands.

6. The database also keeps track of all the Payments made by the Customer:

6.1 Each transaction made has a unique Payment ID.

6.2 Each payment has its mode of payment and the amount to be paid.

7. Amazon database also consist of Staff which work for Customer Care:

   7.1 Each Staff has its own unique Employee ID.

   7.2 Every Employee has an ID, name, email address, phone number.

II. Components of Database Design

To effectively design a database ,we need to have a clear understanding of the entity sets and the relationships of the entity sets in the database .Entity in the context is an object,a component of data.An entity set is collection of similar entities.These entities have different attributes that defines its properties.

The entities and their respectives attributes required are as follows:

1. **STAFF-** As our project is related to the online shopping store we need a huge force of staff for maintenance and effective running of the store.The database keeps the track of the information of the staff working in the amazon store.

Attributes-

- ➢ **Employee Id-** (varchar) e.g ’E001’
- ➢ **Employee name-**(varchar) e.g ‘Arshad Arif’
- ➢ **Employee address -**(varchar) e.g ‘73 , Bastin Drive’
- ➢ **Employee phone number-**(number) e.g ‘'675-8993

2. **CUSTOMER -** Customers are the clients who have done shopping on our store .Every customer has their own unique id and various other attributes.

Attributes-

- ➢ **Customer Id -** (varchar) e.g ‘C001’
- ➢ **Customer Name-**(varchar) e.g ‘Gajanan Rane’
- ➢ **Customer Address-**(varchar) e.g ’Krishnali Complex, Vitthal Mandir Agashi Road’
- ➢ **Customer Phone Number -** (number) e.g ‘8652558944’
- ➢ **Customer Email Id -**(varchar) e.g ‘krishcom13@gmail.com’

3. **SUPPLIER-** Since it is an amazon database we have suppliers which supplies products to our store and then it is available for our customers.

   Attributes-

   - ➢ <u>**Supplier Id**</u> **-** (varchar) e.g ‘S001’
   - ➢ **Supplier Information-**(varchar) e.g ’Shah Shirts’
   - ➢ **Supplier Address -**(varchar) e.g ‘MUMBAI’

4. **PRODUCTS_-** We have various products available in our store for our clients which includes clothing accessories .Every product has an unique product id and other attributes like availability ,brand and discounts.

   Attributes-

   - ➢ <u>**Product Id**</u> -e.g ‘P001’
   - ➢ **Product Name-**(varchar) e.g ‘ SHIRTS’
   - ➢ **Product Cost-** (number) e.g ‘2000’
   - ➢ **Availability -**(boolean) e,g ‘TRUE’
   - ➢ **Discount -**(number) e.g ’10%’
   - ➢ **Brand** -(varchar) e.g ‘MANGO’

5. **ORDERS-**When customers want to buy something he needs to place an order.So we have created an entity order which keeps the track of its details like order-id, order-date and many other details.

   Attributes-

   - ➢ <u>**Order-id**</u> - e.g ‘OR1’
   - ➢ **Order-date**
   - ➢ **Delivery- Date**

6. **PAYMENT -** To buy products from our store customers will do payments so in our database we will keep track of the customers payment record .

Attributes-

- **Payment-id** -e.g ‘PMT1’
- **Payment Type**- (varchar) e.g ‘COD’
- **Amount-** (number) e.g ‘300’

We have listed all the entities which are required for our database .The next step is to list down the **relationships** between different entities .

A **cardinality** notation defines the attributes of the relationship between the entities.Cardinalities can denote that an entity is optional or mandatory.

There are three types of cardinalities;

- A one-to-one relationship-(1:1)
- A one-to-many relationship-(1:M)
- A many-to-many relationship-(M:M)

Relationships and Cardinality

In our amazon database we have five types of relationships that are as follows:

- Entities **Supplier** and **Products** are connected by a relation called **Supplies**.There is a **many-to-many** relationship between these two entities .Where both the sides i.e supplier and product has **total** participation.

- Entities **Staff** and **Customer** are connected by a relation called **Customercare**.There is a **many-to-many** relationship between these two entities .Where Customer side is **partial** participation and Staff has **total** participation.

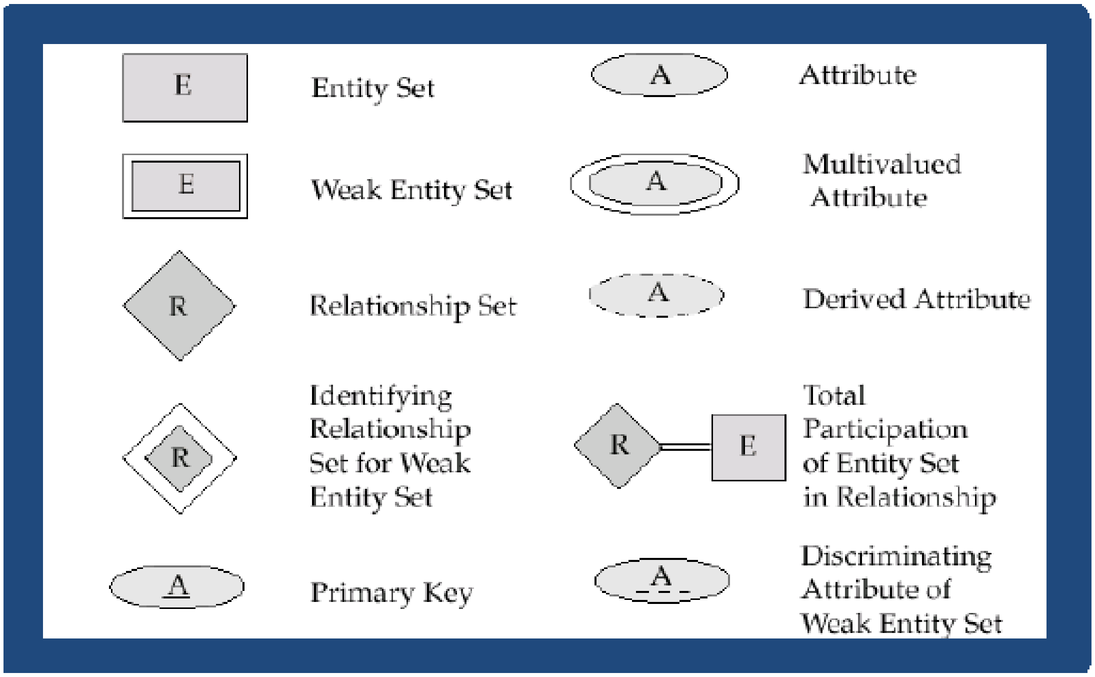
- Entities **Order** and **Product** are connected by a relation called **Cart**.There is a **many-to-many** relationship between these two entities .Where Product side is **total** participation and Order has **total** participation.

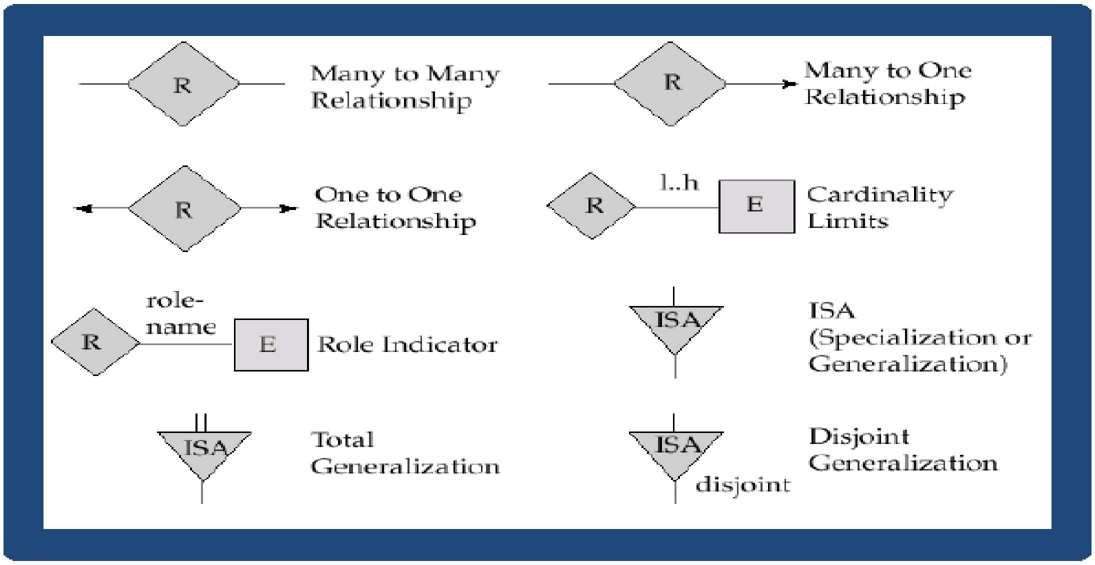
- Entities **Customer** and **Order** are connected by a relation called **Places**.There is a **many-to-one** relationship between these two entities .Where Customer side is **partial** participation and Order has **total** participation.

- Entities **Payment** and **Order** are connected by a relation called **Pays**.There is a **many-to-many** relationship between these two entities .Where Payment side is **total** participation and Order has **total** participation.

III. Entity Relationship Diagram

An E-R diagram provides a visual starting point for a database design .It helps to determine information systems requirements throughout an organization . The main purpose of the E-R diagram is to model a complex database.Concepts of E-R diagram is to be followed while making E-R diagrams such as their conventional symbol methods used in it.

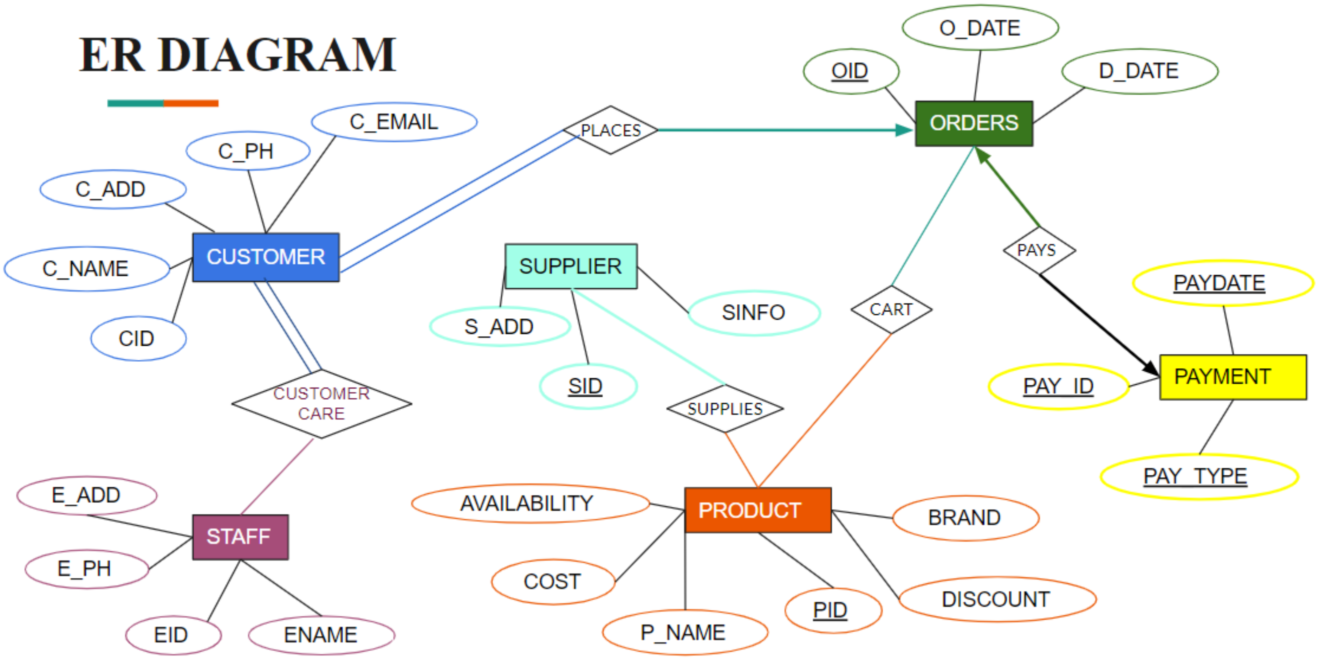

E
Entity Set
A
Attribute
E
Weak Entity Set
A
Multivalued Attribute
R
Relationship Set
A
Derived Attribute
R
Identifying Relationship Set for Weak Entity Set
R
E
Total Participation of Entity Set in Relationship
A
Primary Key
A
Discriminating Attribute of Weak Entity Set



First step to draw the E-R diagram is that we have to list down all the entities and their **primary ke**y for e.g Entity Customer has attributes (CID, C_NAME, C_ADD, C_PH, C_EMAIL) in which CID is the primary key.

Similar steps are to be followed for all the entities -

- PRODUCT (PID, P_NAME, COST, AVAILABILITY, DISCOUNT, BRAND), PID is the primary key.
- SUPPLIER (SID, S_INFO, S_ADD), SID is the primary key.
- STAFF (EID, ENAME, EADD, EPH) , EID is the primary key.
- PAYMENT (PAYID, PAY_TYPE, AMOUNT), PAYID is the primary key.
- ORDERS (OID, O_DATE, D_DATE), OID is the primary key.



IV. Relational Model

CUSTOMER (CID, C_NAME, C_ADD, C_PH, C_EMAIL)

PRODUCT (PID, P_NAME, COST, AVAILABILITY, DISCOUNT, BRAND)

SUPPLIER (SID, SINFO, S_ADD)

PAYMENT (PAYID, PAY_TYPE, PAYDATE)

STAFF (EID, ENAME, EADD, EPH)

ORDERS (OID,CID*, PID*, O_DATE, D_DATE)

SUPPLIES (SID*, PID*)

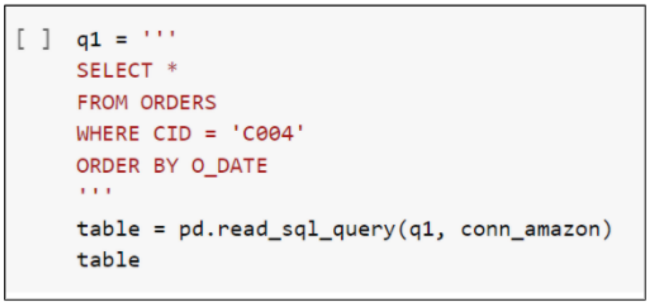
CUSTOMERCARE (EID*, CID*)

CART (OID*, PID*, QUANTITY)

V. SQL Queries

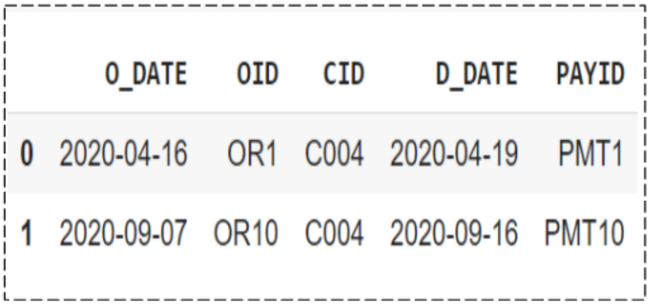
1) Display all the orders placed by Customer with Customer ID 'C004'.

INPUT



```
[ ] q1 = '''
    SELECT *
    FROM ORDERS
    WHERE CID = 'C004'
    ORDER BY O_DATE
    '''
    table = pd.read_sql_query(q1, conn_amazon)
    table
```

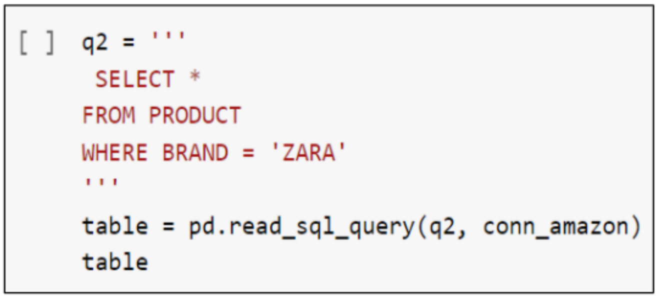
OUTPUT



	O_DATE	OID	CID	D_DATE	PAYID
0	2020-04-16	OR1	C004	2020-04-19	PMT1
1	2020-09-07	OR10	C004	2020-09-16	PMT10

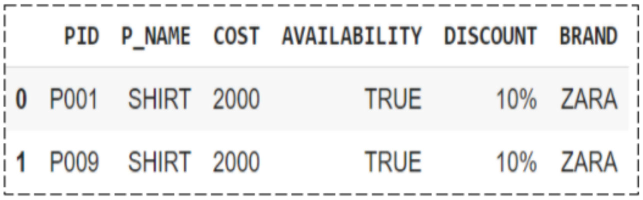
2) Display all the product from the brand ZARA.

INPUT



```
[ ] q2 = '''
     SELECT *
    FROM PRODUCT
    WHERE BRAND = 'ZARA'
    '''
    table = pd.read_sql_query(q2, conn_amazon)
    table
```

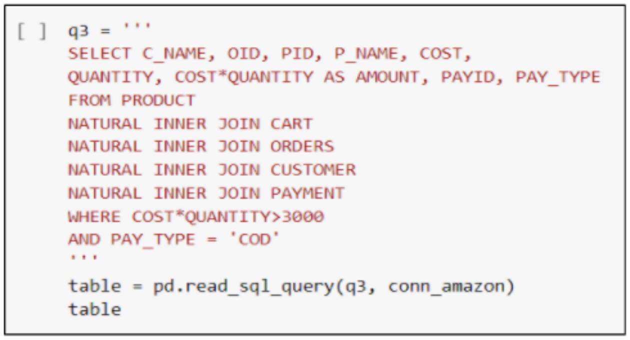
OUTPUT



	PID	P_NAME	COST	AVAILABILITY	DISCOUNT	BRAND
0	P001	SHIRT	2000	TRUE	10%	ZARA
1	P009	SHIRT	2000	TRUE	10%	ZARA

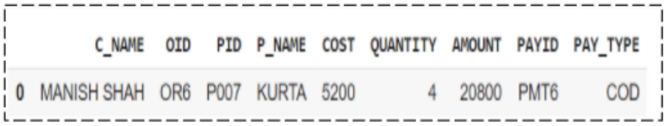
3) Display all the customers who have opted for Cash on Delivery with Amount greater than Rs.3000.

INPUT



```
[ ]  q3 = '''
     SELECT C_NAME, OID, PID, P_NAME, COST,
     QUANTITY, COST*QUANTITY AS AMOUNT, PAYID, PAY_TYPE
     FROM PRODUCT
     NATURAL INNER JOIN CART
     NATURAL INNER JOIN ORDERS
     NATURAL INNER JOIN CUSTOMER
     NATURAL INNER JOIN PAYMENT
     WHERE COST*QUANTITY>3000
     AND PAY_TYPE = 'COD'
     '''
     table = pd.read_sql_query(q3, conn_amazon)
     table
```

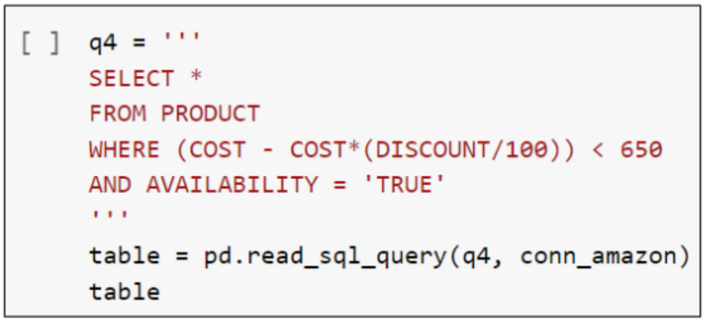
OUTPUT



	C_NAME	OID	PID	P_NAME	COST	QUANTITY	AMOUNT	PAYID	PAY_TYPE
0	MANISH SHAH	OR6	P007	KURTA	5200	4	20800	PMT6	COD

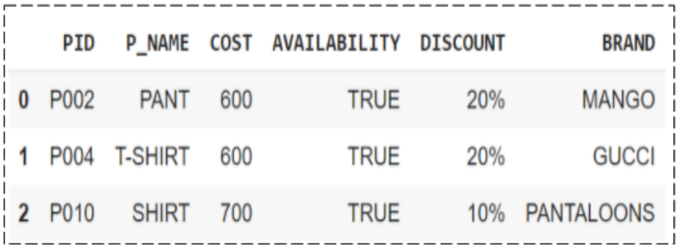
4) Display all the available products which have price less than 650 after applying the discount.

INPUT



```
[ ]  q4 = '''
     SELECT *
     FROM PRODUCT
     WHERE (COST - COST*(DISCOUNT/100)) < 650
     AND AVAILABILITY = 'TRUE'
     '''
     table = pd.read_sql_query(q4, conn_amazon)
     table
```

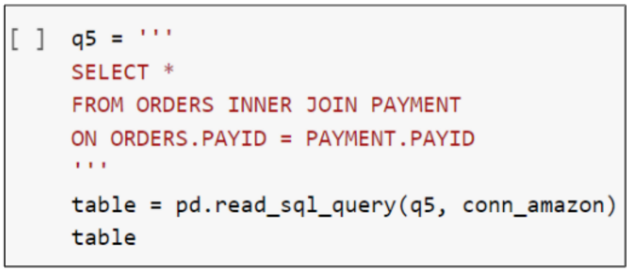
OUTPUT



	PID	P_NAME	COST	AVAILABILITY	DISCOUNT	BRAND
0	P002	PANT	600	TRUE	20%	MANGO
1	P004	T-SHIRT	600	TRUE	20%	GUCCI
2	P010	SHIRT	700	TRUE	10%	PANTALOONS

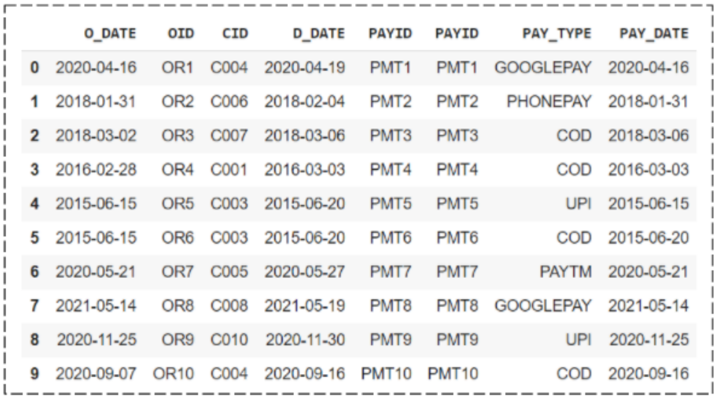
5) Display all the orders and their corresponding payments made based on payment ID

INPUT



```
[ ] q5 = '''
    SELECT *
    FROM ORDERS INNER JOIN PAYMENT
    ON ORDERS.PAYID = PAYMENT.PAYID
    '''
    table = pd.read_sql_query(q5, conn_amazon)
    table
```

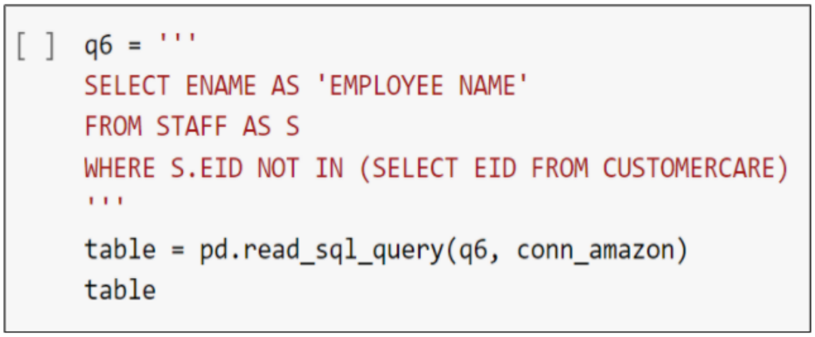
OUTPUT



	O_DATE	OID	CID	D_DATE	PAYID	PAYID	PAY_TYPE	PAY_DATE
0	2020-04-16	OR1	C004	2020-04-19	PMT1	PMT1	GOOGLEPAY	2020-04-16
1	2018-01-31	OR2	C006	2018-02-04	PMT2	PMT2	PHONEPAY	2018-01-31
2	2018-03-02	OR3	C007	2018-03-06	PMT3	PMT3	COD	2018-03-06
3	2016-02-28	OR4	C001	2016-03-03	PMT4	PMT4	COD	2016-03-03
4	2015-06-15	OR5	C003	2015-06-20	PMT5	PMT5	UPI	2015-06-15
5	2015-06-15	OR6	C003	2015-06-20	PMT6	PMT6	COD	2015-06-20
6	2020-05-21	OR7	C005	2020-05-27	PMT7	PMT7	PAYTM	2020-05-21
7	2021-05-14	OR8	C008	2021-05-19	PMT8	PMT8	GOOGLEPAY	2021-05-14
8	2020-11-25	OR9	C010	2020-11-30	PMT9	PMT9	UPI	2020-11-25
9	2020-09-07	OR10	C004	2020-09-16	PMT10	PMT10	COD	2020-09-16

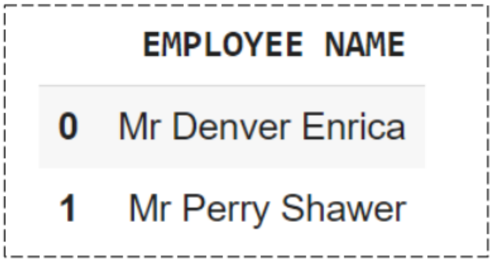
6) Display all the employees who are currently handling no customers

INPUT



```
[ ] q6 = '''
    SELECT ENAME AS 'EMPLOYEE NAME'
    FROM STAFF AS S
    WHERE S.EID NOT IN (SELECT EID FROM CUSTOMERCARE)
    '''
    table = pd.read_sql_query(q6, conn_amazon)
    table
```

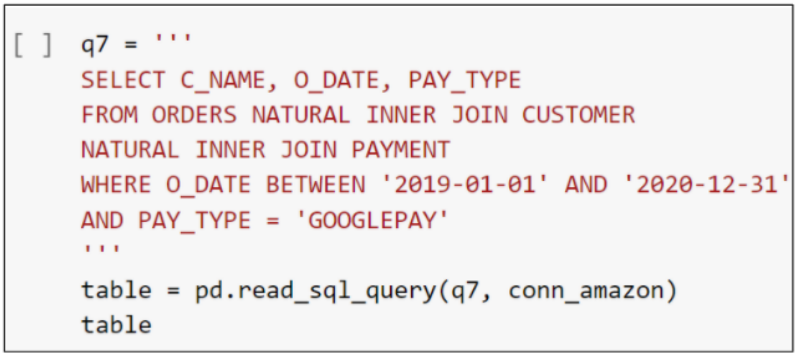
OUTPUT



	EMPLOYEE NAME
0	Mr Denver Enrica
1	Mr Perry Shawer

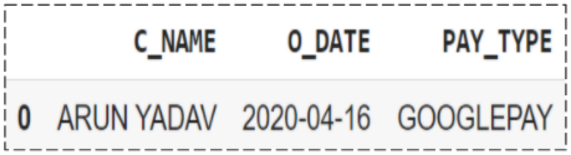
7) Display all the customers who have placed an order between 2019 - 2020 and have made payment through Google Pay

INPUT



```
[ ]  q7 = '''
     SELECT C_NAME, O_DATE, PAY_TYPE
     FROM ORDERS NATURAL INNER JOIN CUSTOMER
     NATURAL INNER JOIN PAYMENT
     WHERE O_DATE BETWEEN '2019-01-01' AND '2020-12-31'
     AND PAY_TYPE = 'GOOGLEPAY'
     '''
     table = pd.read_sql_query(q7, conn_amazon)
     table
```

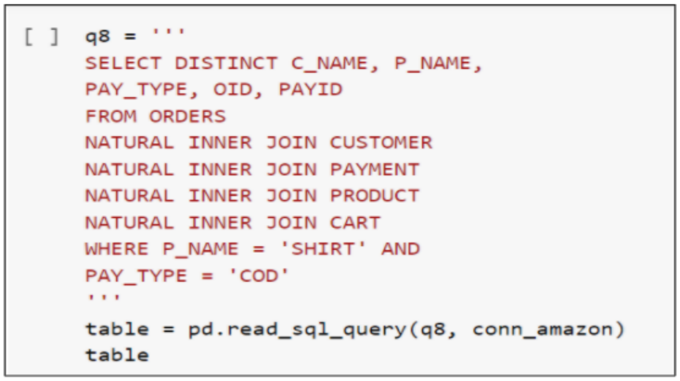
OUTPUT



	C_NAME	O_DATE	PAY_TYPE
0	ARUN YADAV	2020-04-16	GOOGLEPAY

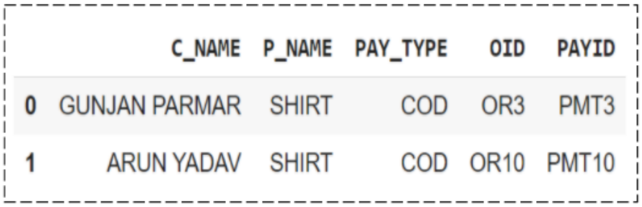
8) Display all the customers who have ordered a shirt and have opted for cash on delivery.

INPUT



```
[ ]  q8 = '''
     SELECT DISTINCT C_NAME, P_NAME,
     PAY_TYPE, OID, PAYID
     FROM ORDERS
     NATURAL INNER JOIN CUSTOMER
     NATURAL INNER JOIN PAYMENT
     NATURAL INNER JOIN PRODUCT
     NATURAL INNER JOIN CART
     WHERE P_NAME = 'SHIRT' AND
     PAY_TYPE = 'COD'
     '''
     table = pd.read_sql_query(q8, conn_amazon)
     table
```

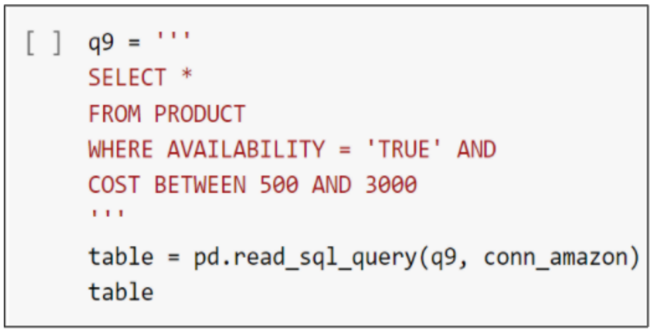
OUTPUT



	C_NAME	P_NAME	PAY_TYPE	OID	PAYID
0	GUNJAN PARMAR	SHIRT	COD	OR3	PMT3
1	ARUN YADAV	SHIRT	COD	OR10	PMT10

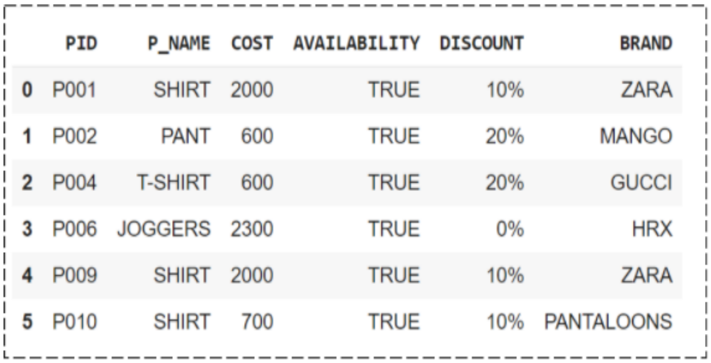
9) Display all the products which are available and have a price between Rs. 500 - 3000

INPUT



```
[ ]  q9 = '''
     SELECT *
     FROM PRODUCT
     WHERE AVAILABILITY = 'TRUE' AND
     COST BETWEEN 500 AND 3000
     '''
     table = pd.read_sql_query(q9, conn_amazon)
     table
```

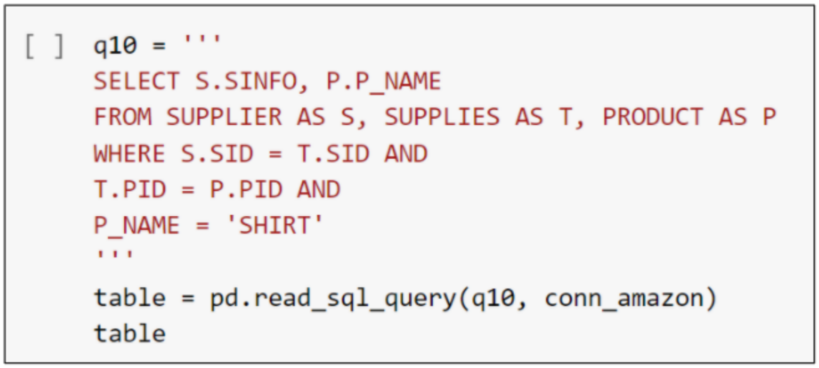
OUTPUT



	PID	P_NAME	COST	AVAILABILITY	DISCOUNT	BRAND
0	P001	SHIRT	2000	TRUE	10%	ZARA
1	P002	PANT	600	TRUE	20%	MANGO
2	P004	T-SHIRT	600	TRUE	20%	GUCCI
3	P006	JOGGERS	2300	TRUE	0%	HRX
4	P009	SHIRT	2000	TRUE	10%	ZARA
5	P010	SHIRT	700	TRUE	10%	PANTALOONS

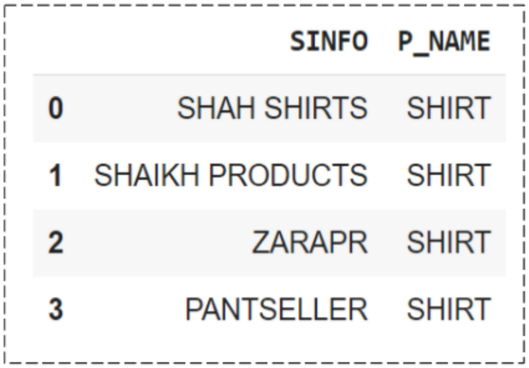
10) Display all the supplier who supply shirt.

INPUT



```
[ ]  q10 = '''
     SELECT S.SINFO, P.P_NAME
     FROM SUPPLIER AS S, SUPPLIES AS T, PRODUCT AS P
     WHERE S.SID = T.SID AND
     T.PID = P.PID AND
     P_NAME = 'SHIRT'
     '''
     table = pd.read_sql_query(q10, conn_amazon)
     table
```

OUTPUT



	SINFO	P_NAME
0	SHAH SHIRTS	SHIRT
1	SHAIKH PRODUCTS	SHIRT
2	ZARAPR	SHIRT
3	PANTSELLER	SHIRT

VI. CONCLUSION

Using SQL and DBMS the following project was implemented. We have made use of Google Colaboratory for simulating the queries. SQLite3 was used as the main primary language for running the queries with the help of Pandas library of Python to simulate the working. Various concepts of SQL and DBMS were used in the project. We have created tables and inserted the corresponding values in the external files and have imported them. With the help of a cursor a connection to the database is established. Thus the Python - SQL integration helped us in implementing the Amazon Database.