

**PROJECT ANALYSIS REPORT**  
**SUBMITTED FOR**  
**DATABASE MANAGEMENT SYSTEM (UCS310)**

**SUBMITTED BY:**

**KASHIKA CHOPRA 102203492**

**KANIKA KUKKAR 102203559**

**RIDDHI SEKHRI 102203598**

**SMRITI SINGH 102203604**



**THAPAR INSTITUTE**  
OF ENGINEERING & TECHNOLOGY  
(Deemed to be University)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY,**  
**PATIALA-147004, PUNJAB**  
**JAN-MAY 2024**

## **ACKNOWLEDGEMENT**

We extend our heartfelt gratitude to all those who have contributed to the completion of this project. However, it would not have been possible without the kind support and help of many individuals and teachers. We would like to extend our thanks to all of them.

We express our sincere appreciation to Dr. Archana for her invaluable guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

Our thanks and appreciations also go to our colleagues for their dedicated efforts in researching, analyzing and designing the database schema and functionalities in developing the project and people who have willingly helped us out with their abilities.

Together, we have all played an integral role in the realization of this e-commerce management project, and for that, we are truly grateful.

**Signature of Faculty**

# **INTRODUCTION**

In an era defined by digital transformation and the rapid evolution of consumer behavior, e-commerce has emerged as a cornerstone of modern business. The seamless integration of technology, logistics, and customer-centric strategies has redefined how businesses engage with their audience and drive growth in the global marketplace.

## **THE E-COMMERCE DATABASE:**

This project is all about making online shopping easy and enjoyable for both businesses and customers. Imagine a world where buying and selling online is not just easy but also fun. That's what our e-commerce management system is all about – making life simpler for businesses and customers alike. From keeping track of products to making sure orders are delivered on time, our system has got it covered. It's like having a super-smart assistant for your online store!

One of the core features of our e-commerce management system is its ability to streamline the entire sales process. From cataloging products and managing stock levels to processing payments securely, our system ensures smooth and efficient transactions, reducing the burden on business owners and enhancing the shopping experience for customers.

## **We first draw up a clear list of requirements for our database:**

- User Authentication and Authorization
- Product Management
- Order Processing
- Shopping Cart
- Payment Gateway Integration
- Customer Relationship Management
- Search and Navigation
- Security and Compliance

There is no requirement of Physical inventory management, Shipping logistics integration, Physical point of sale, Localization and Multilanguage support.

By excluding these features, we keep our project focused on providing essential e-commerce management functionalities while allowing flexibility for users to integrate additional tools or services as needed for their specific requirements.

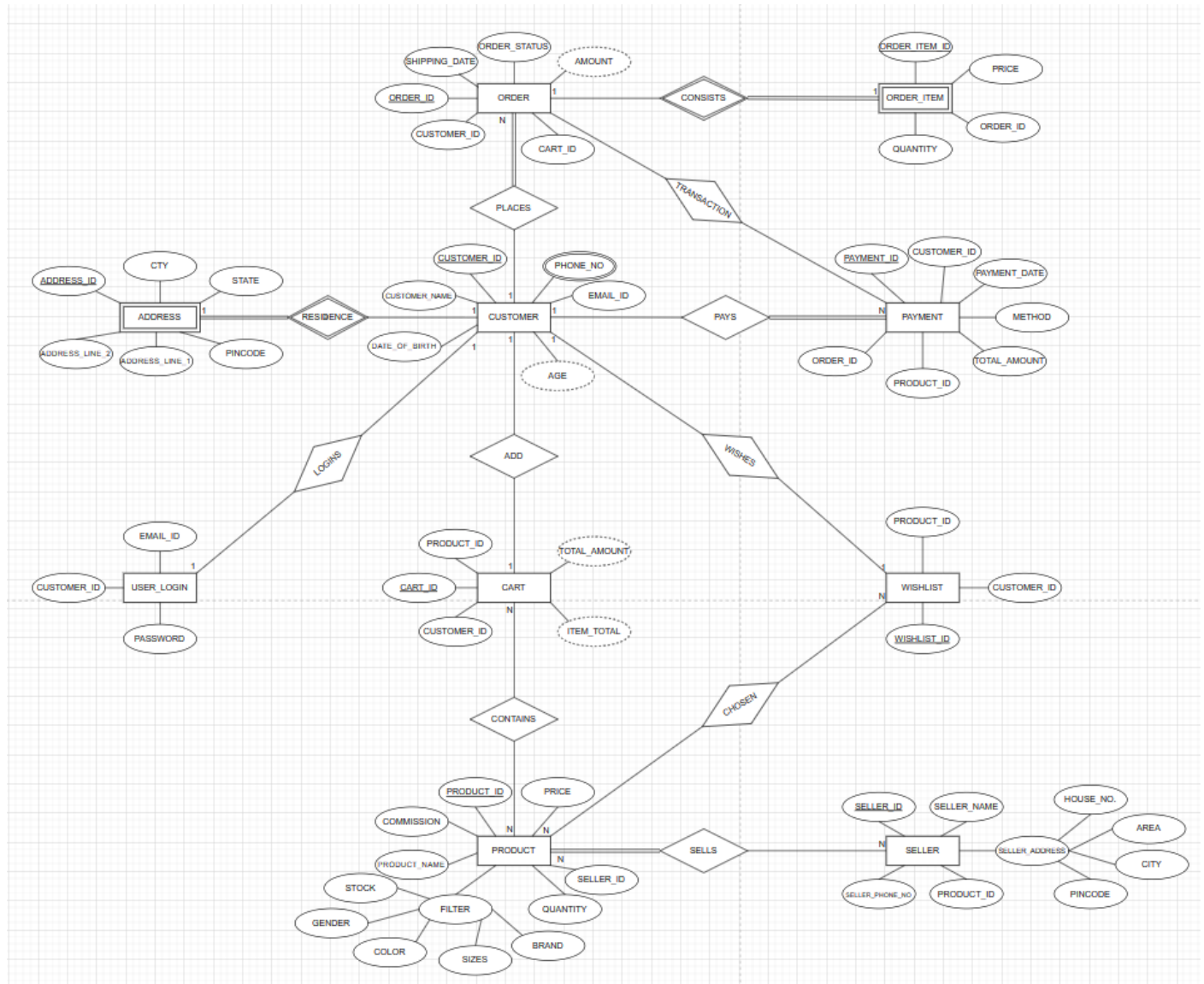
The ER diagram derived from our requirements is shown in below figure. The attributes are straightforward: customer, order, address, payment, wishlist, cart, user\_login, seller, product, filter, order\_item as well as identifiers to uniquely identify each entity.

## **INDEX**

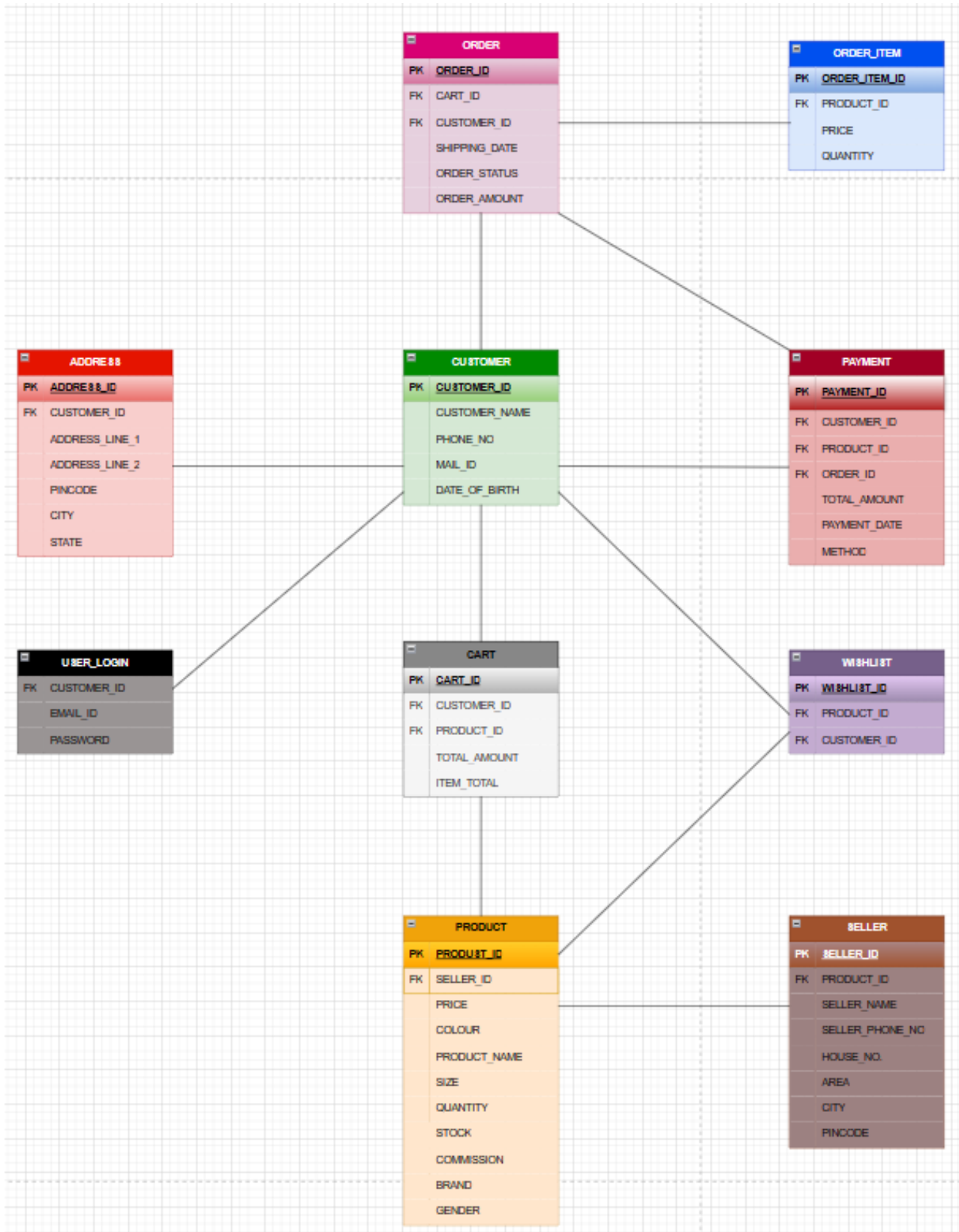
<b>S No.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
1.	ER Diagram	5
2.	ER to Table	6
3.	Tables Used	7
4.	Normalization	17
5.	SQL code and output screenshots	22
6.	PLSQL Queries and output screenshots	44
7.	SQL Queries and output screenshots	48
8.	Conclusion	49
9.	References	50

# ER DIAGRAM

The ER diagram derived from our requirements is shown in below figure. The attributes are straightforward: customer, order, address, payment, wishlist, cart, user\_login, seller, product, filter, order\_item as well as identifiers to uniquely identify each entity.



# ER TO TABLE



## TABLES USED

### 1. CUSTOMER:

This table, part of an e-commerce website's database, stores customer information. It includes fields for unique customer ID, name, phone number, email, and date of birth. The primary key is the customer ID ensuring each entry is distinct, facilitating efficient data management and customer interactions.

COLUMN NAME	DATA TYPE	CONSTRAINTS
CUSTOMER_ID	NUMBER	PRIMARY KEY
CUSTOMER_NAME	VARCHAR	
PHONE_NO.	NUMBER	
MAIL_ID	VARCHAR	
DATE_OF_BIRTH	DATE	

CUSTOMER_ID	CUSTOMER_NAME	DATE_OF_BIRTH	EMAIL_ID
1	Rajesh Kumar	12-APR-85	rajesh.kumar@gmail.com
2	Priya Singh	21-JUL-90	priya.singh@gmail.com
3	Amit Patel	05-SEP-82	amit.patel@gmail.com
4	Divya Sharma	18-MAR-95	divya.sharma@gmail.com
5	Sanjay Gupta	30-NOV-88	sanjay.gupta@gmail.com
6	Sneha Verma	24-JUN-93	sneha.verma@gmail.com
7	Ritu Mehta	14-FEB-80	ritu.mehta@gmail.com
8	Anil Yadav	09-AUG-77	anil.yadav@gmail.com
9	Pooja Malhotra	20-MAY-98	pooja.malhotra@gmail.com
10	Kiran Reddy	17-OCT-84	kiran.reddy@gmail.com
11	Manoj Tiwari	03-DEC-86	manoj.tiwari@gmail.com
12	Neha Kapoor	28-SEP-91	neha.kapoor@gmail.com
13	Vivek Joshi	10-JUL-79	vivek.joshi@gmail.com

## 2. PRODUCT:

This table stores product details for an e-commerce platform, including identifiers, seller information, and item specifics such as type, size, quantity, and price range. It tracks inventory via 'STOCK' and categorizes products by age group, gender, color, and brand. Key constraints ensure data integrity between related tables.

COLUMN NAME	DATA TYPE	CONSTRAINTS
PRODUCT_ID	NUMBER	PRIMARY KEY
SELLER_ID	NUMBER	FOREIGN KEY
PRICE	NUMBER	
COMMISSION	NUMBER	
PRODUCT NAME	VARCHAR	
SIZES	VARCHAR	
QUANTITY	NUMBER	
STOCK	VARCHAR	
GENDER	VARCHAR	
COLOR	VARCHAR	
BRAND	VARCHAR	

PRODUCT_ID	PRODUCT_NAME	PRICE	QUANTITY	COMMISSION	GENDER	SIZES	COLOR	BRAND	STOCK	SELLER_ID
5001	mens t-shirt	999	50	150	male	M	black	adidas	100	1001
5002	womens jeans	1499	30	200	female	L	blue	levis	80	1002
5003	mens polo shirt	899	40	120	male	XL	green	puma	120	1003
5004	womens dress	1999	25	250	female	M	red	zara	60	1004
5005	mens sneakers	2999	20	300	male	L	white	nike	50	1005
5006	womens sandals	799	35	100	female	S	brown	bata	90	1006
5007	mens formal shirt	1299	45	180	male	XXL	blue	van heusen	70	1007
5008	womens leggings	599	40	80	female	M	black	H&M	110	1008
5012	womens sweater	1199	35	150	female	S	white	zara	65	1012
5013	mens jacket	2499	20	250	male	L	black	roadster	55	1013
5014	womens coat	2999	15	300	female	M	brown	H&M	40	1014
5015	mens jeans	1799	40	170	male	XL	blue	levis	75	1015
5016	womens top	699	50	90	female	L	yellow	forever 21	85	1001



### 3. SELLER:

This table, stores seller information for an e-commerce platform. It includes the seller's ID (primary key), product connections (via foreign key on PRODUCT\_ID), name, phone number, and address details such as house number, area, city, and pincode, facilitating management of seller data.

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
SELLER_ID	NUMBER	PRIMARY KEY
PRODUCT_ID	NUMBER	FOREIGN KEY
SELLER_NAME	VARCHAR	
SELLER_PHONE_NO	NUMBER	
HOUSE_NO.	NUMBER	
AREA	VARCHAR	
CITY	VARCHAR	
PINCODE	NUMBER	

SELLER_ID	SELLER_NAME	SELLER_PHONE_NO	HOUSE_NO	AREA	CITY	PINCODE
1001	Fashion Hub	9876543210	123	Main Street	Mumbai	400001
1002	Trendy Wear	8765432109	456	Fashion Avenue	Delhi	110001
1003	Chic Styles	7654321098	789	Fashion Plaza	Bangalore	560001
1004	Glamour Attire	6543210987	234	Fashion Park	Chennai	600001
1005	Urban Trends	5432109876	567	Fashion Street	Kolkata	700001
1006	Casual Closet	4321098765	890	Fashion Lane	Hyderabad	500001
1007	Dress Code	3210987654	901	Fashion Road	Pune	411001
1008	Style Statement	2109876543	234	Fashion Avenue	Ahmedabad	380001
1009	Designer Dreams	1098765432	567	Fashion Street	Jaipur	302001
1010	Trend Setters	1234567890	890	Fashion Lane	Lucknow	226001
1011	Fashion Forward	2345678901	901	Fashion Road	Chandigarh	160001
1012	Elite Elegance	3456789012	123	Fashion Street	Nagpur	440001
1013	Vogue Vault	4567890123	456	Fashion Park	Indore	452001

#### 4. ORDER:

This table, designed for an e-commerce website, stores order details. It includes the order ID as the primary key, with foreign keys for the cart and customer IDs. Other attributes include shipping date, order status, and order amount, essential for managing and tracking customer purchases efficiently.

COLUMN NAME	DATA TYPE	CONSTRAINTS
ORDER_ID	NUMBER	PRIMARY KEY
CART_ID	NUMBER	FOREIGN KEY
CUSTOMER_ID	NUMBER	FOREIGN KEY
SHIPPING_DATE	DATE	
ORDER_STATUS	VARCHAR	
ORDER_AMOUNT	NUMBER	

ORDER_ID	ORDER_STATUS	SHIPPING_DATE	CUSTOMER_ID	CART_ID
1000101	order placed	04-MAY-24	1	10001
2000202	shipped	05-MAY-24	2	20002
3000303	out for delivery	06-MAY-24	3	30003
4000404	delivered	07-MAY-24	4	40004
5000505	order placed	08-MAY-24	5	50005
6000606	shipped	09-MAY-24	6	60006
7000707	out for delivery	10-MAY-24	7	70007
8000808	delivered	11-MAY-24	8	80008

## 5. PAYMENT:

This table records payment details for transactions on an e-commerce platform. Each entry includes a unique payment ID, customer and product IDs linked to other tables, the total amount, payment date, and method. It efficiently tracks and relates payments to customers, orders, and products.

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
PAYMENT_ID	NUMBER	PRIMARY KEY
CUSTOMER_ID	NUMBER	FOREIGN KEY
PRODUCT_ID	NUMBER	FOREIGN KEY
ORDER_ID	NUMBER	FOREIGN KEY
TOTAL_AMOUNT	NUMBER	
PAYMENT_DATE	DATE	
METHOD	VARCHAR	

PAYMENT_ID	METHODS	CUSTOMER_ID	PRODUCT_ID	ORDER_ITEM_ID	PAYMENT_DATE
1563	UPI	1	5001	900101	05-SEP-23
1564	credit card	2	5002	903601	08-SEP-23
1565	debit card	3	5003	900102	12-SEP-23
1567	wallet	5	5005	900301	20-SEP-23
1568	UPI	6	5006	900302	25-SEP-23
1569	credit card	7	5007	900401	28-SEP-23
1570	debit card	8	5008	900501	02-OCT-23
1574	credit card	12	5012	901002	15-OCT-23
1575	debit card	13	5013	901101	18-OCT-23
1577	wallet	15	5015	901202	25-OCT-23
1578	UPI	16	5016	901301	28-OCT-23
1579	credit card	17	5017	901501	02-NOV-23
1580	debit card	18	5018	901502	05-NOV-23

## 6. CART:

This table represents a shopping cart in an e-commerce database. Each row details a cart instance with fields for cart ID (primary key), customer and product IDs (foreign keys), total amount payable, and item-total, facilitating transaction tracking and customer-product linkage in the overall shopping process.

COLUMN NAME	DATA TYPE	CONSTRAINTS
CART_ID	NUMBER	PRIMARY KEY
CUSTOMER_ID	NUMBER	FOREIGN KEY
PRODUCT_ID	NUMBER	FOREIGN KEY
TOTAL_AMOUNT	NUMBER	
ITEM_TOTAL	NUMBER	

CART_ID	CUSTOMER_ID	PRODUCT_ID
10001	1	5001
20002	2	5002
30003	3	5003
40004	4	5004
50005	5	5005
60006	6	5006
70007	7	5007
80008	8	5008
12012	12	5012
13013	13	5013
14014	14	5014
15015	15	5015
16016	16	5016

## 7. ADDRESS:

This table, part of an e-commerce website's database, stores addresses for customers. It includes primary key ADDRESS\_ID and foreign key CUSTOMER\_ID for relational integrity. Other fields capture address details such as lines, pin code, city, and state, facilitating accurate order deliveries and customer management.

COLUMN NAME	DATA TYPE	CONSTRAINTS
ADDRESS_ID	NUMBER	PRIMARY KEY
CUSTOMER_ID	NUMBER	FOREIGN KEY
ADDRESS_LINE_1	VARCHAR	
ADDRESS_LINE_2	VARCHAR	
PINCODE	NUMBER	
CITY	VARCHAR	
STATE	VARCHAR	

ADDRESS_ID	CITY	S_STATE	PINCODE	ADDRESS_LINE1	ADDRESS_LINE2	CUSTOMER_ID
2000	Mumbai	Maharashtra	400001	123, ABC Street	Near XYZ Park	1
2001	Delhi	Delhi	110001	456, XYZ Road	Opposite ABC Mall	2
2002	Kolkata	West Bengal	700001	789, PQR Avenue	Adjacent to LMN Hospital	3
2003	Chennai	Tamil Nadu	600001	1011, LMN Lane	Behind PQR Market	4
2004	Bangalore	Karnataka	560001	1213, RST Road	Next to EFG Garden	5
2005	Hyderabad	Telangana	500001	1415, UVW Street	Near GHI School	6
2006	Pune	Maharashtra	411001	1617, HIJ Avenue	Beside UVW Park	7
2007	Ahmedabad	Gujarat	380001	1819, EFG Lane	Opposite HIJ Mall	8
2008	Jaipur	Rajasthan	302001	2021, STU Road	Adjacent to VWX Hospital	9
2009	Lucknow	Uttar Pradesh	226001	2223, WXY Avenue	Near RST Market	10
2010	Patna	Bihar	800001	2425, OPQ Street	Next to STU Garden	11
2011	Noida	Uttar Pradesh	201301	2627, XYZ Road	Opposite ABC Mall	12
2012	Gurgaon	Haryana	122001	2829, PQR Avenue	Adjacent to LMN Hospital	13

## 8. USER LOGIN:

This table, part of an e-commerce database, stores customer authentication details. Columns include 'CUSTOMER\_ID' (foreign key linking to the primary customer table), 'EMAIL\_ID', and 'PASSWORD'. It ensures each customer's email is unique and credentials are secure for login purposes.

COLUMN NAME	DATA TYPE	CONSTRAINTS
CUSTOMER_ID	NUMBER	FOREIGN KEY
EMAIL_ID	VARCHAR	
PASSWORD	VARCHAR	

EMAIL	PASSWORD	CUSTOMER_ID
rajesh.kumar@gmail.com	rajesh001	1
priya.singh@gmail.com	priya002	2
amit.patel@gmail.com	amit003	3
divya.sharma@gmail.com	divya004	4
sanjay.gupta@gmail.com	sanjay005	5
sneha.verma@gmail.com	sneha006	6
ritu.mehta@gmail.com	ritu007	7
anil.yadav@gmail.com	anil008	8
pooja.malhotra@gmail.com	pooja009	9
kiran.reddy@gmail.com	kiran010	10
manoj.tiwari@gmail.com	manoj011	11
neha.kapoor@gmail.com	neha012	12
vivek.joshi@gmail.com	vivek013	13

## 9. ORDER ITEM:

This table tracks individual items within orders on an e-commerce platform. Each record, uniquely identified by 'ORDER\_ITEM\_ID', links to a product via 'PRODUCT\_ID' (foreign key), and specifies the 'PRICE' and 'QUANTITY' purchased. The primary key ensures unique entries, while the foreign key maintains referential integrity with the products table.

COLUMN_NAME	DATA_TYPE	CONSTRAINTS
ORDER_ITEM_ID	NUMBER	PRIMARY KEY
PRODUCT_ID	NUMBER	FOREIGN KEY
PRICE	NUMBER	
QUANTITY	NUMBER	

ORDER_ITEM_ID	PRICE	QUANTITY	CART_ID
900101	1000	2	10001
903601	600	1	60006
900102	750	3	20002
900201	1200	1	30003
900301	500	2	40004
900302	800	1	50005
900401	1500	1	60006
900501	2000	2	70007
900601	600	1	80008
900901	1100	1	12012
901001	750	2	13013
901002	800	1	14014
901101	2000	1	15015

## 10. WISHLIST:

The table described above is the 'Wishlist' table for an e-commerce website. It stores user-specified products for potential future purchases. 'WISHLIST\_ID' is the primary key, uniquely identifying each Wishlist entry. 'PRODUCT\_ID' and 'CUSTOMER\_ID' are foreign keys linking to the Products and Customers tables respectively.

COLUMN NAME	DATA TYPE	CONSTRAINTS
WISHLIST_ID	NUMBER	PRIMARY KEY
PRODUCT_ID	NUMBER	FOREIGN KEY
CUSTOMER_ID	NUMBER	FOREIGN KEY

WISHLIST_ID	CUSTOMER_ID	PRODUCT_ID
111	1	5001
112	2	5003
113	3	5005
114	4	5007
117	7	5013
118	8	5015
119	9	5017
121	11	5021
123	13	5025
124	14	5027
125	15	5029
127	17	5033
128	18	5035



# **NORMALIZATION**

**Normalization** is a process of organizing the data in the database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

## **1. First normal form (1NF)**

As per the rule of the first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

## **2. Second normal form (2NF)**

A table is said to be in 2NF if both the following conditions hold:

- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of the table.

An attribute that is not part of any candidate key is known as a non-prime attribute.

## **3. Third Normal form (3NF)**

A table design is said to be in 3NF if both the following conditions hold:

- Table must be in 2NF
- Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as a non-prime attribute.

In other words, 3NF can be explained like this: A table is in 3NF if it is in 2NF, and for each functional dependency  $X \rightarrow Y$  at least one of the following conditions holds:

- X is a super key of the table
- Y is a prime attribute of the table

An attribute that is a part of one of the candidate keys is known as a prime attribute.

## 1. FIRST NORMAL FORM (1NF):

1. We have composite attributes in two tables which are PRODUCT TABLE and SELLER TABLE. So, we have applied 1NF to normalize it.

1. PRODUCT TABLE:

In this table, FILTER is the composite attribute with GENDER, SIZES, COLOR, BRAND, and STOCK sub-attributes. And it is being normalized.

PRODUCT_ID	PRODUCT_NAME	PRICE	QUANTITY	COMMISSION	GENDER	SIZES	COLOR	BRAND	STOCK	SELLER_ID
5001	mens t-shirt	999	50	150	male	M	black	adidas	100	1001
5002	womens jeans	1499	30	200	female	L	blue	levis	80	1002
5003	mens polo shirt	899	40	120	male	XL	green	puma	120	1003
5004	womens dress	1999	25	250	female	M	red	zara	60	1004
5005	mens sneakers	2999	20	300	male	L	white	nike	50	1005

2. SELLER TABLE:

In this table, SELLER\_ADDRESS is the composite attribute with HOUSE\_NO, AREA, CITY, PINCODE sub-attributes. And it is being normalized.

SELLER_ID	SELLER_NAME	SELLER_PHONE_NO	HOUSE_NO	AREA	CITY	PINCODE
1001	Fashion Hub	9876543210	123	Main Street	Mumbai	400001
1002	Trendy Wear	8765432109	456	Fashion Avenue	Delhi	110001
1003	Chic Styles	7654321098	789	Fashion Plaza	Bangalore	560001
1004	Glamour Attire	6543210987	234	Fashion Park	Chennai	600001
1005	Urban Trends	5432109876	567	Fashion Street	Kolkata	700001

2. We have multivalued attributes in the CUSTOMER TABLE. So, we have applied 1NF to normalize it.

CUSTOMER_ID	CUSTOMER_NAME	DATE_OF_BIRTH	EMAIL_ID
1	Rajesh Kumar	12-APR-85	rajesh.kumar@gmail.com
2	Priya Singh	21-JUL-90	priya.singh@gmail.com
3	Amit Patel	05-SEP-82	amit.patel@gmail.com
4	Divya Sharma	18-MAR-95	divya.sharma@gmail.com
5	Sanjay Gupta	30-NOV-88	sanjay.gupta@gmail.com

PHONE_NUMBER	CUSTOMER_ID
9876543210	1
1234567890	2
2345678901	3
3456789012	4
4567890123	5

In this table combination of CUSTOMER\_ID and PHONE\_NUMBER. PHONE\_NUMBER is the primary key and CUSTOMER\_ID is the foreign key. It has been normalized from the customer table.

## 2. SECOND NORMAL FORM (2NF):

Both the "SELLER" and "PRODUCT" tables are in 1NF as they contain atomic values in each cell, and there are no repeating groups.

### 1. SELLER TABLE:

In the "SELLER" table, all the non-prime attributes ("SELLER\_NAME," "SELLER\_PHONE\_NO," "HOUSE\_NO," "AREA," "CITY," "PINCODE") are fully functionally dependent on the primary key "SELLER\_ID."

SELLER_ID	SELLER_NAME	SELLER_PHONE_NO	HOUSE_NO	AREA	CITY	PINCODE
1001	Fashion Hub	9876543210	123	Main Street	Mumbai	400001
1002	Trendy Wear	8765432109	456	Fashion Avenue	Delhi	110001
1003	Chic Styles	7654321098	789	Fashion Plaza	Bangalore	560001
1004	Glamour Attire	6543210987	234	Fashion Park	Chennai	600001

### 2. PRODUCT TABLE:

In the "PRODUCT" table, all the non-prime attributes ("PRODUCT\_NAME," "PRICE," "QUANTITY," "COMMISSION," "GENDER," "SIZES," "COLOR," "BRAND," "STOCK") are fully functionally dependent on the primary key "PRODUCT\_ID."

PRODUCT_ID	PRODUCT_NAME	PRICE	QUANTITY	COMMISSION	GENDER	SIZES	COLOR	BRAND	STOCK	SELLER_ID
5001	mens t-shirt	999	50	150	male	M	black	adidas	100	1001
5002	womens jeans	1499	30	200	female	L	blue	levis	80	1002
5003	mens polo shirt	899	40	120	male	XL	green	puma	120	1003
5004	womens dress	1999	25	250	female	M	red	zara	60	1004
5005	mens sneakers	2999	20	300	male	L	white	nike	50	1005

### 3. THIRD NORMAL FORM (3NF):

Both tables:

- Do not show any transitive dependencies among the non-primary attributes.
- All non-primary attributes directly depend on the primary key.

Thus, both tables meet the criteria for 3NF.

#### 1. CUSTOMER TABLE:

CUSTOMER_ID	CUSTOMER_NAME	DATE_OF_BIRTH	EMAIL_ID
1	Rajesh Kumar	12-APR-85	rajesh.kumar@gmail.com
2	Priya Singh	21-JUL-90	priya.singh@gmail.com
3	Amit Patel	05-SEP-82	amit.patel@gmail.com
4	Divya Sharma	18-MAR-95	divya.sharma@gmail.com
5	Sanjay Gupta	30-NOV-88	sanjay.gupta@gmail.com

#### 2. ADDRESS TABLE:

ADDRESS_ID	CITY	S_STATE	PINCODE	ADDRESS_LINE1	ADDRESS_LINE2	CUSTOMER_ID
2000	Mumbai	Maharashtra	400001	123, ABC Street	Near XYZ Park	1
2001	Delhi	Delhi	110001	456, XYZ Road	Opposite ABC Mall	2
2002	Kolkata	West Bengal	700001	789, PQR Avenue	Adjacent to LMN Hospital	3
2003	Chennai	Tamil Nadu	600001	1011, LMN Lane	Behind PQR Market	4
2004	Bangalore	Karnataka	560001	1213, RST Road	Next to EFG Garden	5

# **SQL: MAIN SOURCE CODE**

## **1. CUSTOMER TABLE:**

```
CREATE TABLE customer(  
    customer_id number(10) primary key,  
    customer_name varchar(20) not null,  
    date_of_birth date,  
    email_id varchar(30)  
);  
  
INSERT INTO customer VALUES (1, 'Rajesh Kumar', TO_DATE('1985-04-12', 'YYYY-MM-DD'),  
'rajesh.kumar@gmail.com');  
  
INSERT INTO customer VALUES (2, 'Priya Singh', TO_DATE('1990-07-21', 'YYYY-MM-DD'),  
'priya.singh@gmail.com');  
  
INSERT INTO customer VALUES (3, 'Amit Patel', TO_DATE('1982-09-05', 'YYYY-MM-DD'),  
'amit.patel@gmail.com');  
  
INSERT INTO customer VALUES (4, 'Divya Sharma', TO_DATE('1995-03-18', 'YYYY-MM-DD'),  
'divya.sharma@gmail.com');  
  
INSERT INTO customer VALUES (5, 'Sanjay Gupta', TO_DATE('1988-11-30', 'YYYY-MM-DD'),  
'sanjay.gupta@gmail.com');  
  
INSERT INTO customer VALUES (6, 'Sneha Verma', TO_DATE('1993-06-24', 'YYYY-MM-DD'),  
'sneha.verma@gmail.com');  
  
INSERT INTO customer VALUES (7, 'Ritu Mehta', TO_DATE('1980-02-14', 'YYYY-MM-DD'),  
'ritu.mehta@gmail.com');  
  
INSERT INTO customer VALUES (8, 'Anil Yadav', TO_DATE('1977-08-09', 'YYYY-MM-DD'),  
'anil.yadav@gmail.com');  
  
INSERT INTO customer VALUES (9, 'Pooja Malhotra', TO_DATE('1998-05-20', 'YYYY-MM-DD'),  
'pooja.malhotra@gmail.com');  
  
INSERT INTO customer VALUES (10, 'Kiran Reddy', TO_DATE('1984-10-17', 'YYYY-MM-DD'),  
'kiran.reddy@gmail.com');  
  
INSERT INTO customer VALUES (11, 'Manoj Tiwari', TO_DATE('1986-12-03', 'YYYY-MM-DD'),  
'manoj.tiwari@gmail.com');  
  
INSERT INTO customer VALUES (12, 'Neha Kapoor', TO_DATE('1991-09-28', 'YYYY-MM-DD'),  
'neha.kapoor@gmail.com');  
  
INSERT INTO customer VALUES (13, 'Vivek Joshi', TO_DATE('1979-07-10', 'YYYY-MM-DD'),  
'vivek.joshi@gmail.com');  
  
INSERT INTO customer VALUES (14, 'Anjali Singh', TO_DATE('1983-05-15', 'YYYY-MM-DD'),  
'anjali.singh@gmail.com');
```

INSERT INTO customer VALUES (15, 'Suresh Sharma', TO\_DATE('1978-08-22', 'YYYY-MM-DD'), 'suresh.sharma@gmail.com');

INSERT INTO customer VALUES (16, 'Deepika Patel', TO\_DATE('1996-02-19', 'YYYY-MM-DD'), 'deepika.patel@gmail.com');

INSERT INTO customer VALUES (17, 'Rahul Gupta', TO\_DATE('1981-01-07', 'YYYY-MM-DD'), 'rahul.gupta@gmail.com');

INSERT INTO customer VALUES (18, 'Riya Malhotra', TO\_DATE('1994-04-23', 'YYYY-MM-DD'), 'riya.malhotra@gmail.com');

INSERT INTO customer VALUES (19, 'Alok Singh', TO\_DATE('1975-11-09', 'YYYY-MM-DD'), 'alok.singh@gmail.com');

INSERT INTO customer VALUES (20, 'Aditi Verma', TO\_DATE('1992-06-06', 'YYYY-MM-DD'), 'aditi.verma@gmail.com');

INSERT INTO customer VALUES (21, 'Suman Yadav', TO\_DATE('1987-03-30', 'YYYY-MM-DD'), 'suman.yadav@gmail.com');

INSERT INTO customer VALUES (22, 'Ravi Kumar', TO\_DATE('1984-10-02', 'YYYY-MM-DD'), 'ravi.kumar@gmail.com');

INSERT INTO customer VALUES (23, 'Preeti Sharma', TO\_DATE('1976-09-14', 'YYYY-MM-DD'), 'preeti.sharma@gmail.com');

INSERT INTO customer VALUES (24, 'Akash Gupta', TO\_DATE('1993-08-07', 'YYYY-MM-DD'), 'akash.gupta@gmail.com');

INSERT INTO customer VALUES (25, 'Ananya Singh', TO\_DATE('1988-07-28', 'YYYY-MM-DD'), 'ananya.singh@gmail.com');

INSERT INTO customer VALUES (26, 'Vikas Malhotra', TO\_DATE('1980-04-11', 'YYYY-MM-DD'), 'vikas.malhotra@gmail.com');

INSERT INTO customer VALUES (27, 'Juhi Patel', TO\_DATE('1997-12-16', 'YYYY-MM-DD'), 'juhi.patel@gmail.com');

INSERT INTO customer VALUES (28, 'Rajeshwari Singh', TO\_DATE('1979-03-25', 'YYYY-MM-DD'), 'rajeshwari.singh@gmail.com');

INSERT INTO customer VALUES (29, 'Kunal Sharma', TO\_DATE('1982-11-20', 'YYYY-MM-DD'), 'kunal.sharma@gmail.com');

INSERT INTO customer VALUES (30, 'Shreya Gupta', TO\_DATE('1995-10-08', 'YYYY-MM-DD'), 'shreya.gupta@gmail.com');

Live SQL

SQL Worksheet

```

1 create table customer(
2     customer_id number(10) primary key,
3     customer_name varchar(20) not null,
4     date_of_birth date,
5     email_id varchar(30)
6 );
7 INSERT INTO customer VALUES (1, 'Rajesh Kumar', TO_DATE('1985-04-12', 'YYYY-MM-DD'), 'rajesh.kumar@gmail.com');
8 INSERT INTO customer VALUES (2, 'Priya Singh', TO_DATE('1990-07-21', 'YYYY-MM-DD'), 'priya.singh@gmail.com');
9 INSERT INTO customer VALUES (3, 'Amit Patel', TO_DATE('1982-09-05', 'YYYY-MM-DD'), 'amit.patel@gmail.com');
10 INSERT INTO customer VALUES (4, 'Divya Sharma', TO_DATE('1995-03-18', 'YYYY-MM-DD'), 'divya.sharma@gmail.com');
11 INSERT INTO customer VALUES (5, 'Sanjay Gupta', TO_DATE('1988-11-30', 'YYYY-MM-DD'), 'sanjay.gupta@gmail.com');
12 INSERT INTO customer VALUES (6, 'Sneha Verma', TO_DATE('1993-06-24', 'YYYY-MM-DD'), 'sneha.verma@gmail.com');
13 INSERT INTO customer VALUES (7, 'Ritu Mehta', TO_DATE('1980-02-14', 'YYYY-MM-DD'), 'ritu.mehta@gmail.com');
14 INSERT INTO customer VALUES (8, 'Anil Yadav', TO_DATE('1977-08-09', 'YYYY-MM-DD'), 'anil.yadav@gmail.com');
15 INSERT INTO customer VALUES (9, 'Pooja Malhotra', TO_DATE('1998-05-20', 'YYYY-MM-DD'), 'pooja.malhotra@gmail.com');
16 INSERT INTO customer VALUES (10, 'Kiran Reddy', TO_DATE('1984-10-17', 'YYYY-MM-DD'), 'kiran.reddy@gmail.com');
17 INSERT INTO customer VALUES (11, 'Mehar Tiwari', TO_DATE('1986-12-02', 'YYYY-MM-DD'), 'mehar.tiwari@gmail.com');

```

CUSTOMER_ID	CUSTOMER_NAME	DATE_OF_BIRTH	EMAIL_ID
1	Rajesh Kumar	12-APR-85	rajesh.kumar@gmail.com
2	Priya Singh	21-JUL-90	priya.singh@gmail.com
3	Amit Patel	05-SEP-82	amit.patel@gmail.com
4	Divya Sharma	18-MAR-95	divya.sharma@gmail.com

## 2. PHONE\_NUMBER TABLE:

```

create table phone_number(
    phone_number number(10),
    customer_id references customer(customer_id)
);
INSERT INTO phone_number VALUES (9876543210, 1);
INSERT INTO phone_number VALUES (1234567890, 2);
INSERT INTO phone_number VALUES (2345678901, 3);
INSERT INTO phone_number VALUES (3456789012, 4);
INSERT INTO phone_number VALUES (4567890123, 5);
INSERT INTO phone_number VALUES (5678901234, 6);
INSERT INTO phone_number VALUES (6789012345, 7);
INSERT INTO phone_number VALUES (7890123456, 8);
INSERT INTO phone_number VALUES (8901234567, 9);
INSERT INTO phone_number VALUES (9012345678, 10);
INSERT INTO phone_number VALUES (1357924680, 11);
INSERT INTO phone_number VALUES (2468013579, 12);
INSERT INTO phone_number VALUES (3579246801, 13);
INSERT INTO phone_number VALUES (4680135792, 14);
INSERT INTO phone_number VALUES (5792468013, 15);
INSERT INTO phone_number VALUES (6901357924, 16);

```



```
INSERT INTO phone_number VALUES (8013579246, 17);
INSERT INTO phone_number VALUES (9124680135, 18);
INSERT INTO phone_number VALUES (2468013579, 19);
INSERT INTO phone_number VALUES (3579246801, 20);
INSERT INTO phone_number VALUES (4680135792, 21);
INSERT INTO phone_number VALUES (5792468013, 22);
INSERT INTO phone_number VALUES (6901357924, 23);
INSERT INTO phone_number VALUES (8013579246, 24);
INSERT INTO phone_number VALUES (9124680135, 25);
INSERT INTO phone_number VALUES (1234567890, 26);
INSERT INTO phone_number VALUES (2345678901, 27);
INSERT INTO phone_number VALUES (3456789012, 28);
INSERT INTO phone_number VALUES (4567890123, 29);
INSERT INTO phone_number VALUES (5678901234, 30);
INSERT INTO phone_number VALUES (6789012345, 1);
INSERT INTO phone_number VALUES (7890123456, 2);
INSERT INTO phone_number VALUES (8901234567, 3);
INSERT INTO phone_number VALUES (9012345678, 4);
INSERT INTO phone_number VALUES (5432109876, 5);
INSERT INTO phone_number VALUES (6543210987, 6);
INSERT INTO phone_number VALUES (7654321098, 7);
INSERT INTO phone_number VALUES (8765432109, 8);
INSERT INTO phone_number VALUES (9876543210, 9);
INSERT INTO phone_number VALUES (8901234567, 10);
```

Live SQL

SQL Worksheet

```

40 --PHONE NUMBER:
41 create table phone_number(
42     phone_number number(10),
43     customer_id references customer(customer_id)
44 );
45 INSERT INTO phone_number VALUES (9876543210, 1);
46 INSERT INTO phone_number VALUES (1234567890, 2);
47 INSERT INTO phone_number VALUES (2345678901, 3);
48 INSERT INTO phone_number VALUES (3456789012, 4);
49 INSERT INTO phone_number VALUES (4567890123, 5);
50 INSERT INTO phone_number VALUES (5678901234, 6);
51 INSERT INTO phone_number VALUES (6789012345, 7);
52 INSERT INTO phone_number VALUES (7890123456, 8);
53 INSERT INTO phone_number VALUES (8901234567, 9);
54 INSERT INTO phone_number VALUES (9012345678, 10);
55 INSERT INTO phone_number VALUES (1357924680, 11);
56 TRUNCATE TABLE phone_number;

```

PHONE_NUMBER	CUSTOMER_ID
9876543210	1
1234567890	2
2345678901	3
3456789012	4

### 3. ADDRESS TABLE:

CREATE TABLE address(

address\_id NUMBER(6),

city VARCHAR2(20),

s\_state VARCHAR2(20),

pincode NUMBER(6) NOT NULL,

address\_line1 VARCHAR2(30),

address\_line2 VARCHAR2(30),

customer\_id NUMBER(10) REFERENCES customer(customer\_id)

);

INSERT INTO address VALUES (2000, 'Mumbai', 'Maharashtra', 400001, '123, ABC Street', 'Near XYZ Park', 1);

INSERT INTO address VALUES (2001, 'Delhi', 'Delhi', 110001, '456, XYZ Road', 'Opposite ABC Mall', 2);

INSERT INTO address VALUES (2002, 'Kolkata', 'West Bengal', 700001, '789, PQR Avenue', 'Adjacent to LMN Hospital', 3);

INSERT INTO address VALUES (2003, 'Chennai', 'Tamil Nadu', 600001, '1011, LMN Lane', 'Behind PQR Market', 4);

INSERT INTO address VALUES (2004, 'Bangalore', 'Karnataka', 560001, '1213, RST Road', 'Next to EFG Garden', 5);

INSERT INTO address VALUES (2005, 'Hyderabad', 'Telangana', 500001, '1415, UVW Street', 'Near GHI School', 6);

INSERT INTO address VALUES (2006, 'Pune', 'Maharashtra', 411001, '1617, HIJ Avenue', 'Beside UVW Park', 7);

INSERT INTO address VALUES (2007, 'Ahmedabad', 'Gujarat', 380001, '1819, EFG Lane', 'Opposite HIJ Mall', 8);

INSERT INTO address VALUES (2008, 'Jaipur', 'Rajasthan', 302001, '2021, STU Road', 'Adjacent to VWX Hospital', 9);

INSERT INTO address VALUES (2009, 'Lucknow', 'Uttar Pradesh', 226001, '2223, WXY Avenue', 'Near RST Market', 10);

INSERT INTO address VALUES (2010, 'Patna', 'Bihar', 800001, '2425, OPQ Street', 'Next to STU Garden', 11);

INSERT INTO address VALUES (2011, 'Noida', 'Uttar Pradesh', 201301, '2627, XYZ Road', 'Opposite ABC Mall', 12);

INSERT INTO address VALUES (2012, 'Gurgaon', 'Haryana', 122001, '2829, PQR Avenue', 'Adjacent to LMN Hospital', 13);

INSERT INTO address VALUES (2013, 'Chandigarh', 'Punjab', 160001, '3031, LMN Lane', 'Behind PQR Market', 14);

INSERT INTO address VALUES (2014, 'Indore', 'Madhya Pradesh', 452001, '3233, RST Road', 'Next to EFG Garden', 15);

INSERT INTO address VALUES (2015, 'Bhopal', 'Madhya Pradesh', 462001, '3435, UVW Street', 'Near GHI School', 16);

INSERT INTO address VALUES (2016, 'Kanpur', 'Uttar Pradesh', 208001, '3637, HIJ Avenue', 'Beside UVW Park', 17);

INSERT INTO address VALUES (2017, 'Nagpur', 'Maharashtra', 440001, '3839, EFG Lane', 'Opposite HIJ Mall', 18);

INSERT INTO address VALUES (2018, 'Visakhapatnam', 'Andhra Pradesh', 530001, '4041, STU Road', 'Adjacent to VWX Hospital', 19);

INSERT INTO address VALUES (2019, 'Thane', 'Maharashtra', 400601, '4243, WXY Avenue', 'Near RST Market', 20);

INSERT INTO address VALUES (2020, 'Agra', 'Uttar Pradesh', 282001, '4445, OPQ Street', 'Next to STU Garden', 21);

INSERT INTO address VALUES (2021, 'Varanasi', 'Uttar Pradesh', 221001, '4647, XYZ Road', 'Opposite ABC Mall', 22);

INSERT INTO address VALUES (2022, 'Srinagar', 'Jammu and Kashmir', 190001, '4849, PQR Avenue', 'Adjacent to LMN Hospital', 23);

INSERT INTO address VALUES (2023, 'Ghaziabad', 'Uttar Pradesh', 201001, '5051, LMN Lane', 'Behind PQR Market', 24);

INSERT INTO address VALUES (2024, 'Amritsar', 'Punjab', 143001, '5253, RST Road', 'Next to EFG Garden', 25);

INSERT INTO address VALUES (2025, 'Allahabad', 'Uttar Pradesh', 211001, '5455, UVW Street', 'Near GHI School', 26);

INSERT INTO address VALUES (2026, 'Ranchi', 'Jharkhand', 834001, '5657, HIJ Avenue', 'Beside UVW Park', 27);

INSERT INTO address VALUES (2027, 'Howrah', 'West Bengal', 711101, '5859, EFG Lane', 'Opposite HIJ Mall', 28);

INSERT INTO address VALUES (2028, 'Coimbatore', 'Tamil Nadu', 641001, '6061, STU Road', 'Adjacent to VWX Hospital', 29);

INSERT INTO address VALUES (2029, 'Vadodara', 'Gujarat', 390001, '6263, WXY Avenue', 'Near RST Market', 30);

INSERT INTO address VALUES (2030, 'Rajkot', 'Gujarat', 360001, '6465, OPQ Street', 'Next to STU Garden', 1);

INSERT INTO address VALUES (2031, 'Kochi', 'Kerala', 682001, '6667, XYZ Road', 'Opposite ABC Mall', 2);

INSERT INTO address VALUES (2032, 'Thiruvananthapuram', 'Kerala', 695001, '6869, PQR Avenue', 'Adjacent to LMN Hospital', 3);

INSERT INTO address VALUES (2033, 'Kollam', 'Kerala', 691001, '7071, LMN Lane', 'Behind PQR Market', 4);

INSERT INTO address VALUES (2034, 'Kozhikode', 'Kerala', 673001, '7273, RST Road', 'Next to EFG Garden', 5);

INSERT INTO address VALUES (2035, 'Thrissur', 'Kerala', 680001, '7475, UVW Street', 'Near GHI School', 6);

INSERT INTO address VALUES (2036, 'Alappuzha', 'Kerala', 688001, '7677, HIJ Avenue', 'Beside UVW Park', 7);

INSERT INTO address VALUES (2037, 'Palakkad', 'Kerala', 678001, '7879, EFG Lane', 'Opposite HIJ Mall', 8);

INSERT INTO address VALUES (2038, 'Malappuram', 'Kerala', 676001, '8081, STU Road', 'Adjacent to VWX Hospital', 9);

INSERT INTO address VALUES (2039, 'Kannur', 'Kerala', 670001, '8283, WXY Avenue', 'Near RST Market', 10);

Live SQL

Feedback Help rsekhri\_be22@thapar.edu

Home SQL Worksheet Clear Find Actions Save Run

SQL Worksheet

My Session Schema Quick SQL My Scripts My Tutorials Code Library

```

86 --ADDRESS:
87
88 CREATE TABLE address(
89     address_id NUMBER(6),
90     city VARCHAR2(20),
91     s_state VARCHAR2(20),
92     pincode NUMBER(6) NOT NULL,
93     address_line1 VARCHAR2(30),
94     address_line2 VARCHAR2(30),
95     customer_id NUMBER(10) REFERENCES customer(customer_id)
96 );
97
98 INSERT INTO address VALUES (2000, 'Mumbai', 'Maharashtra', 400001, '123, ABC Street', 'Near XYZ Park', 1);
99 INSERT INTO address VALUES (2001, 'Delhi', 'Delhi', 110001, '456, XYZ Road', 'Opposite ABC Mall', 2);
100 INSERT INTO address VALUES (2002, 'Kolkata', 'West Bengal', 700001, '789, PQR Avenue', 'Adjacent to LMN Hospital', 3);
101 INSERT INTO address VALUES (2003, 'Chennai', 'Tamil Nadu', 600001, '1011, LMN Lane', 'Behind PQR Market', 4);
102 INSERT INTO address VALUES (2004, 'Bangalore', 'Karnataka', 560001, '1213, RST Road', 'Next to EFG Garden', 5);

```

ADDRESS_ID	CITY	S_STATE	PINCODE	ADDRESS_LINE1	ADDRESS_LINE2	CUSTOMER_ID
2000	Mumbai	Maharashtra	400001	123, ABC Street	Near XYZ Park	1
2001	Delhi	Delhi	110001	456, XYZ Road	Opposite ABC Mall	2
2002	Kolkata	West Bengal	700001	789, PQR Avenue	Adjacent to LMN Hospital	3
2003	Chennai	Tamil Nadu	600001	1011, LMN Lane	Behind PQR Market	4

Activate Windows

#### 4. SELLER TABLE:

CREATE TABLE seller(

seller\_id number(5) primary key,

seller\_name varchar2(20),

seller\_phone\_no number(10),

house\_no number(5),

area varchar2(20),

city varchar2(30),

pincode number(6) not null

);

INSERT INTO seller VALUES (1001, 'Fashion Hub', 9876543210, 123, 'Main Street', 'Mumbai', 400001);

INSERT INTO seller VALUES (1002, 'Trendy Wear', 8765432109, 456, 'Fashion Avenue', 'Delhi', 110001);

INSERT INTO seller VALUES (1003, 'Chic Styles', 7654321098, 789, 'Fashion Plaza', 'Bangalore', 560001);

INSERT INTO seller VALUES (1004, 'Glamour Attire', 6543210987, 234, 'Fashion Park', 'Chennai', 600001);

INSERT INTO seller VALUES (1005, 'Urban Trends', 5432109876, 567, 'Fashion Street', 'Kolkata', 700001);

INSERT INTO seller VALUES (1006, 'Casual Closet', 4321098765, 890, 'Fashion Lane', 'Hyderabad', 500001);

INSERT INTO seller VALUES (1007, 'Dress Code', 3210987654, 901, 'Fashion Road', 'Pune', 411001);

INSERT INTO seller VALUES (1008, 'Style Statement', 2109876543, 234, 'Fashion Avenue', 'Ahmedabad', 380001);

INSERT INTO seller VALUES (1009, 'Designer Dreams', 1098765432, 567, 'Fashion Street', 'Jaipur', 302001);

INSERT INTO seller VALUES (1010, 'Trend Setters', 1234567890, 890, 'Fashion Lane', 'Lucknow', 226001);

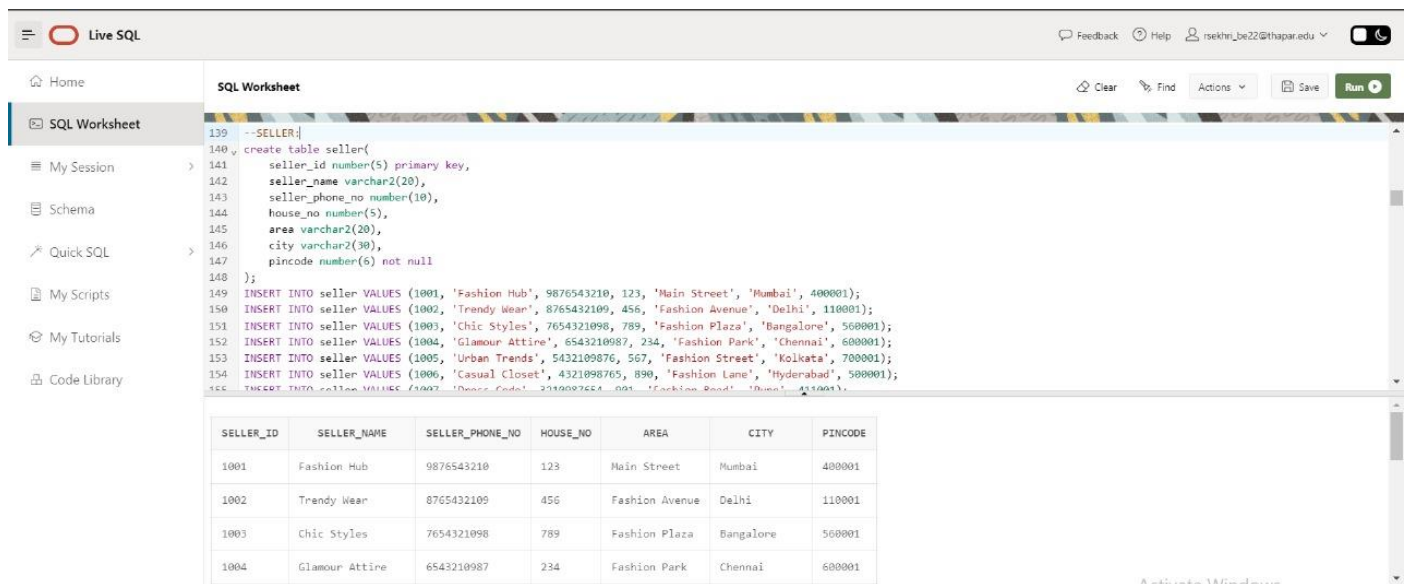
INSERT INTO seller VALUES (1011, 'Fashion Forward', 2345678901, 901, 'Fashion Road', 'Chandigarh', 160001);

INSERT INTO seller VALUES (1012, 'Elite Elegance', 3456789012, 123, 'Fashion Street', 'Nagpur', 440001);

INSERT INTO seller VALUES (1013, 'Vogue Vault', 4567890123, 456, 'Fashion Park', 'Indore', 452001);

INSERT INTO seller VALUES (1014, 'Stylish Creations', 5678901234, 789, 'Fashion Plaza', 'Bhopal', 462001);

INSERT INTO seller VALUES (1015, 'Trendy Threads', 6789012345, 890, 'Fashion Lane', 'Visakhapatnam', 530001);



The screenshot shows a web-based SQL editor interface. On the left is a sidebar with navigation options: Home, SQL Worksheet (selected), My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main area is titled 'SQL Worksheet' and contains SQL code for creating a 'seller' table and inserting 15 records. Below the code, a table view displays the first four rows of data.

```
139 --SELLER|
140 create table seller(
141     seller_id number(5) primary key,
142     seller_name varchar2(20),
143     seller_phone_no number(10),
144     house_no number(5),
145     area varchar2(20),
146     city varchar2(30),
147     pincode number(6) not null
148 );
149 INSERT INTO seller VALUES (1001, 'Fashion Hub', 9876543210, 123, 'Main Street', 'Mumbai', 400001);
150 INSERT INTO seller VALUES (1002, 'Trendy Wear', 8765432109, 456, 'Fashion Avenue', 'Delhi', 110001);
151 INSERT INTO seller VALUES (1003, 'Chic Styles', 7654321098, 789, 'Fashion Plaza', 'Bangalore', 560001);
152 INSERT INTO seller VALUES (1004, 'Glamour Attire', 6543210987, 234, 'Fashion Park', 'Chennai', 600001);
153 INSERT INTO seller VALUES (1005, 'Urban Trends', 5432109876, 567, 'Fashion Street', 'Kolkata', 700001);
154 INSERT INTO seller VALUES (1006, 'Casual Closet', 4321098765, 890, 'Fashion Lane', 'Hyderabad', 500001);
155 INSERT INTO seller VALUES (1007, 'Designer Dreams', 3210987654, 091, 'Fashion Road', 'Jaipur', 302001);
```

SELLER_ID	SELLER_NAME	SELLER_PHONE_NO	HOUSE_NO	AREA	CITY	PINCODE
1001	Fashion Hub	9876543210	123	Main Street	Mumbai	400001
1002	Trendy Wear	8765432109	456	Fashion Avenue	Delhi	110001
1003	Chic Styles	7654321098	789	Fashion Plaza	Bangalore	560001
1004	Glamour Attire	6543210987	234	Fashion Park	Chennai	600001

## 5. PRODUCT TABLE:

CREATE TABLE product(

product\_id NUMBER(10) PRIMARY KEY,

product\_name VARCHAR2(30),

price NUMBER(10) NOT NULL,

quantity NUMBER(3),

commission NUMBER(10),

gender varchar2(6) check (gender in('male','female')),

sizes varchar2(5) check (sizes in ('XS','S','M','L','XL','XXL','3XL','4XL','5XL')),

color varchar2(20) check (color in ('black','white','blue','red','yellow','pink','green','beige','golden','silver','lavender','purple',

```

'orange','multi','brown')),
brand varchar2(30),
stock number(5),
seller_id NUMBER(10) REFERENCES seller(seller_id)
);
INSERT INTO product VALUES (5001, 'mens t-shirt', 999, 50, 150, 'male','M', 'black', 'adidas', 100, 1001);
INSERT INTO product VALUES (5002, 'womens jeans', 1499, 30, 200,'female', 'L', 'blue', 'levis', 80, 1002);
INSERT INTO product VALUES (5003, 'mens polo shirt', 899, 40, 120,'male', 'XL', 'green', 'puma', 120, 1003);
INSERT INTO product VALUES (5004, 'womens dress', 1999, 25, 250, 'female','M', 'red', 'zara', 60, 1004);
INSERT INTO product VALUES (5005, 'mens sneakers', 2999, 20, 300,'male', 'L', 'white', 'nike', 50, 1005);
INSERT INTO product VALUES (5006, 'womens sandals', 799, 35, 100,'female', 'S', 'brown', 'bata', 90, 1006);
INSERT INTO product VALUES (5007, 'mens formal shirt', 1299, 45, 180, 'male','XXL', 'blue', 'van heusen', 70, 1007);
INSERT INTO product VALUES (5008, 'womens leggings', 599, 40, 80, 'female','M', 'black', 'H&M', 110, 1008);
INSERT INTO product VALUES (5009, 'mens shorts', 699, 55, 90, 'male','XL', 'grey', 'HRX', 95, 1009);
INSERT INTO product VALUES (5010, 'womens skirt', 899, 30, 120, 'L', 'pink', 'forever 21', 75, 1010);
INSERT INTO product VALUES (5011, 'mens hoodie', 1499, 25, 200,'male', 'M', 'grey', 'gap', 85, 1011);
INSERT INTO product VALUES (5012, 'womens sweater', 1199, 35, 150,'female', 'S', 'white', 'zara', 65, 1012);
INSERT INTO product VALUES (5013, 'mens jacket', 2499, 20, 250, 'male','L', 'black', 'roadster', 55, 1013);
INSERT INTO product VALUES (5014, 'womens coat', 2999, 15, 300, 'female','M', 'brown', 'H&M', 40, 1014);
INSERT INTO product VALUES (5015, 'mens jeans', 1799, 40, 170,'male', 'XL', 'blue', 'levis', 75, 1015);
INSERT INTO product VALUES (5016, 'womens top', 699, 50, 90,'female', 'L', 'yellow', 'forever 21', 85, 1001);
INSERT INTO product VALUES (5017, 'mens chinos', 1299, 45, 120,'male', 'XXL', 'beige', 'allen solly', 60, 1002);
INSERT INTO product VALUES (5018, 'womens blouse', 899, 30, 100,'female', 'M', 'purple', 'van heusen', 70, 1003);
INSERT INTO product VALUES (5019, 'mens sweatshirt', 999, 35, 150, 'male','XL', 'navy', 'gap', 80, 1004);
INSERT INTO product VALUES (5020, 'womens jumpsuit', 1599, 25, 200,'female', 'S', 'black', 'H&M', 45, 1005);

```

INSERT INTO product VALUES (5021, 'mens polo t-shirt', 799, 40, 100,'male', 'L', 'green', 'puma', 90, 1006);

INSERT INTO product VALUES (5022, 'womens cardigan', 1199, 30, 120,'female', 'M', 'blue', 'zara', 55, 1007);

INSERT INTO product VALUES (5023, 'mens formal trouser', 1299, 35, 150,'male', 'XXL', 'grey', 'van heusen', 70, 1008);

INSERT INTO product VALUES (5024, 'womens scarf', 499, 50, 80, 'female','L', 'pink', 'H&M', 100, 1009);

INSERT INTO product VALUES (5025, 'mens shirt', 999, 45, 120,'male', 'XL', 'white', 'levis', 65, 1010);

INSERT INTO product VALUES (5026, 'womens hoodie', 1499, 25, 200,'female', 'M', 'black', 'gap', 55, 1011);

INSERT INTO product VALUES (5027, 'mens vest', 599, 50, 80,'male', 'L', 'blue', 'nike', 85, 1012);

INSERT INTO product VALUES (5028, 'womens jacket', 1999, 20, 250, 'female','S', 'brown', 'zara', 45, 1013);

INSERT INTO product VALUES (5029, 'mens sweatpants', 1099, 30, 100, 'male','XL', 'black', 'HRX', 70, 1014);

INSERT INTO product VALUES (5030, 'womens leg warmers', 299, 60, 50,'female', 'M', 'grey', 'H&M', 110, 1015);

INSERT INTO product VALUES (5031, 'mens cap', 499, 55, 80, 'male','L', 'navy', 'puma', 75, 1001);

INSERT INTO product VALUES (5032, 'womens sunglasses', 799, 40, 100, 'female','S', 'black', 'ray-ban', 90, 1002);

INSERT INTO product VALUES (5033, 'mens belt', 599, 50, 80, 'male','L', 'brown', 'levis', 85, 1003);

INSERT INTO product VALUES (5034, 'womens earrings', 299, 60, 50, 'female','M', 'silver', 'zara', 110, 1004);

INSERT INTO product VALUES (5035, 'mens watch', 1999, 25, 200,'male', 'L', 'black', 'casio', 50, 1005);

INSERT INTO product VALUES (5036, 'womens shrug', 999, 35, 150, 'female','S', 'gold', 'tanishq', 80, 1006);

INSERT INTO product VALUES (5037, 'mens coat', 1299, 40, 100, 'male','XL', 'brown', 'hidesign', 120, 1007);

INSERT INTO product VALUES (5038, 'womens leggings', 1499, 30, 120,'female', 'M', 'red', 'baggit', 60, 1008);

INSERT INTO product VALUES (5039, 'mens turtle neck', 1999, 20, 250,'male', 'XXL', 'black', 'wildcraft', 55, 1009);

INSERT INTO product VALUES (5040, 'womens sweater', 999, 45, 120,'female', 'L', 'blue', 'lavie', 65, 1010);



Live SQL

Feedback Help rsekhri\_be22@thapar.edu

Home SQL Worksheet Clear Find Actions Save Run

SQL Worksheet

```

166 CREATE TABLE product(
167     product_id NUMBER(10) PRIMARY KEY,
168     product_name VARCHAR2(30),
169     price NUMBER(10) NOT NULL,
170     quantity NUMBER(3),
171     commission NUMBER(10),
172     gender varchar2(6) check (gender in ('male','female')),
173     sizes varchar2(5) check (sizes in ('XS','S','M','L','XL','XXL','3XL','4XL','5XL')),
174     color varchar2(20) check (color in ('black','white','blue','red','yellow','pink','green','beige','golden','silver','lavender','purple','orange','multi','brown')),
175     brand varchar2(30),
176     stock number(5),
177     seller_id NUMBER(10) REFERENCES seller(seller_id)
178 );
179 INSERT INTO product VALUES (5001, 'mens t-shirt', 999, 50, 150, 'male', 'M', 'black', 'adidas', 100, 1001);
180 INSERT INTO product VALUES (5002, 'womens jeans', 1499, 30, 200, 'female', 'L', 'blue', 'levis', 80, 1002);
181 INSERT INTO product VALUES (5003, 'mens polo shirt', 899, 40, 120, 'male', 'XL', 'green', 'puma', 120, 1003);
182 INSERT INTO product VALUES (5004, 'womens dress', 1999, 25, 250, 'female', 'M', 'red', 'zara', 60, 1004);

```

PRODUCT_ID	PRODUCT_NAME	PRICE	QUANTITY	COMMISSION	GENDER	SIZES	COLOR	BRAND	STOCK	SELLER_ID
5001	mens t-shirt	999	50	150	male	M	black	adidas	100	1001
5002	womens jeans	1499	30	200	female	L	blue	levis	80	1002
5003	mens polo shirt	899	40	120	male	XL	green	puma	120	1003
5004	womens dress	1999	25	250	female	M	red	zara	60	1004

Activate Windows

## 6. CART TABLE:

CREATE TABLE cart(

cart\_id NUMBER(10) PRIMARY KEY,

customer\_id NUMBER(10) REFERENCES customer(customer\_id),

product\_id NUMBER(10) REFERENCES product(product\_id)

);

INSERT INTO cart VALUES (10001, 1, 5001);

INSERT INTO cart VALUES (20002, 2, 5002);

INSERT INTO cart VALUES (30003, 3, 5003);

INSERT INTO cart VALUES (40004, 4, 5004);

INSERT INTO cart VALUES (50005, 5, 5005);

INSERT INTO cart VALUES (60006, 6, 5006);

INSERT INTO cart VALUES (70007, 7, 5007);

INSERT INTO cart VALUES (80008, 8, 5008);

INSERT INTO cart VALUES (90009, 9, 5009);

INSERT INTO cart VALUES (10010, 10, 5010);

INSERT INTO cart VALUES (11011, 11, 5011);

INSERT INTO cart VALUES (12012, 12, 5012);

INSERT INTO cart VALUES (13013, 13, 5013);

INSERT INTO cart VALUES (14014, 14, 5014);

INSERT INTO cart VALUES (15015, 15, 5015);

```

INSERT INTO cart VALUES (16016, 16, 5016);
INSERT INTO cart VALUES (17017, 17, 5017);
INSERT INTO cart VALUES (18018, 18, 5018);
INSERT INTO cart VALUES (19019, 19, 5019);
INSERT INTO cart VALUES (20020, 20, 5020);
INSERT INTO cart VALUES (21021, 21, 5021);
INSERT INTO cart VALUES (22022, 22, 5022);
INSERT INTO cart VALUES (23023, 23, 5023);
INSERT INTO cart VALUES (24024, 24, 5024);
INSERT INTO cart VALUES (25025, 25, 5025);
INSERT INTO cart VALUES (26001, 26, 5003);
INSERT INTO cart VALUES (27002, 27, 5007);
INSERT INTO cart VALUES (28003, 28, 5015);
INSERT INTO cart VALUES (29004, 29, 5020);
INSERT INTO cart VALUES (30005, 30, 5023);

```

Live SQL

SQL Worksheet

Clear Find Actions Save Run

```

220 --CART:
221 CREATE TABLE cart(
222     cart_id NUMBER(10) PRIMARY KEY,
223     customer_id NUMBER(10) REFERENCES customer(customer_id),
224     product_id NUMBER(10) REFERENCES product(product_id)
225 );
226 INSERT INTO cart VALUES (10001, 1, 5001);
227 INSERT INTO cart VALUES (20002, 2, 5002);
228 INSERT INTO cart VALUES (30003, 3, 5003);
229 INSERT INTO cart VALUES (40004, 4, 5004);
230 INSERT INTO cart VALUES (50005, 5, 5005);
231 INSERT INTO cart VALUES (60006, 6, 5006);
232 INSERT INTO cart VALUES (70007, 7, 5007);
233 INSERT INTO cart VALUES (80008, 8, 5008);
234 INSERT INTO cart VALUES (90009, 9, 5009);
235 INSERT INTO cart VALUES (10010, 10, 5010);
236 INSERT INTO cart VALUES (11011, 11, 5011);

```

CART_ID	CUSTOMER_ID	PRODUCT_ID
10001	1	5001
20002	2	5002
30003	3	5003
40004	4	5004

Activate Windows

## 7. ORDERS TABLE:

CREATE TABLE orders(

order\_id NUMBER(10),

order\_status VARCHAR2(30) CHECK (order\_status IN ('order placed', 'shipped', 'out for delivery', 'delivered')),

shipping\_date DATE,

customer\_id NUMBER(10) REFERENCES customer(customer\_id),

cart\_id NUMBER(10) REFERENCES cart(cart\_id)

);

INSERT INTO orders VALUES (1000101, 'order placed', TO\_DATE('2024-05-04', 'YYYY-MM-DD'), 1, 10001);

INSERT INTO orders VALUES (2000202, 'shipped', TO\_DATE('2024-05-05', 'YYYY-MM-DD'), 2, 20002);

INSERT INTO orders VALUES (3000303, 'out for delivery', TO\_DATE('2024-05-06', 'YYYY-MM-DD'), 3, 30003);

INSERT INTO orders VALUES (4000404, 'delivered', TO\_DATE('2024-05-07', 'YYYY-MM-DD'), 4, 40004);

INSERT INTO orders VALUES (5000505, 'order placed', TO\_DATE('2024-05-08', 'YYYY-MM-DD'), 5, 50005);

INSERT INTO orders VALUES (6000606, 'shipped', TO\_DATE('2024-05-09', 'YYYY-MM-DD'), 6, 60006);

INSERT INTO orders VALUES (7000707, 'out for delivery', TO\_DATE('2024-05-10', 'YYYY-MM-DD'), 7, 70007);

INSERT INTO orders VALUES (8000808, 'delivered', TO\_DATE('2024-05-11', 'YYYY-MM-DD'), 8, 80008);

INSERT INTO orders VALUES (9000909, 'order placed', TO\_DATE('2024-05-12', 'YYYY-MM-DD'), 9, 90009);

INSERT INTO orders VALUES (10010010, 'shipped', TO\_DATE('2024-05-13', 'YYYY-MM-DD'), 10, 10010);

INSERT INTO orders VALUES (11001111, 'out for delivery', TO\_DATE('2024-05-14', 'YYYY-MM-DD'), 11, 11011);

INSERT INTO orders VALUES (12001212, 'delivered', TO\_DATE('2024-05-15', 'YYYY-MM-DD'), 12, 12012);

INSERT INTO orders VALUES (13001313, 'order placed', TO\_DATE('2024-05-16', 'YYYY-MM-DD'), 13, 13013);

INSERT INTO orders VALUES (14001414, 'shipped', TO\_DATE('2024-05-17', 'YYYY-MM-DD'), 14, 14014);

INSERT INTO orders VALUES (15001515, 'out for delivery', TO\_DATE('2024-05-18', 'YYYY-MM-DD'), 15, 15015);

INSERT INTO orders VALUES (16001616, 'delivered', TO\_DATE('2024-05-19', 'YYYY-MM-DD'), 16, 16016);

INSERT INTO orders VALUES (17001717, 'order placed', TO\_DATE('2024-05-20', 'YYYY-MM-DD'), 17, 17017);

INSERT INTO orders VALUES (18001818, 'shipped', TO\_DATE('2024-05-21', 'YYYY-MM-DD'), 18, 18018);

INSERT INTO orders VALUES (19001919, 'out for delivery', TO\_DATE('2024-05-22', 'YYYY-MM-DD'), 19, 19019);

INSERT INTO orders VALUES (20002020, 'delivered', TO\_DATE('2024-05-23', 'YYYY-MM-DD'), 20, 20020);

INSERT INTO orders VALUES (21002121, 'order placed', TO\_DATE('2024-05-24', 'YYYY-MM-DD'), 21, 21021);

INSERT INTO orders VALUES (22002222, 'shipped', TO\_DATE('2024-05-25', 'YYYY-MM-DD'), 22, 22022);

INSERT INTO orders VALUES (23002323, 'out for delivery', TO\_DATE('2024-05-26', 'YYYY-MM-DD'), 23, 23023);

INSERT INTO orders VALUES (24002424, 'delivered', TO\_DATE('2024-05-27', 'YYYY-MM-DD'), 24, 24024);

INSERT INTO orders VALUES (25002525, 'order placed', TO\_DATE('2024-05-28', 'YYYY-MM-DD'), 25, 25025);

INSERT INTO orders VALUES (2600126, 'shipped', TO\_DATE('2024-05-29', 'YYYY-MM-DD'), 26, 26001);

INSERT INTO orders VALUES (27002727, 'out for delivery', TO\_DATE('2024-05-30', 'YYYY-MM-DD'), 27, 27002);

INSERT INTO orders VALUES (28002828, 'delivered', TO\_DATE('2024-05-31', 'YYYY-MM-DD'), 28, 28003);

INSERT INTO orders VALUES (29002929, 'order placed', TO\_DATE('2024-06-01', 'YYYY-MM-DD'), 29, 29004);

INSERT INTO orders VALUES (30003030, 'shipped', TO\_DATE('2024-06-02', 'YYYY-MM-DD'), 30, 30005);

INSERT INTO orders VALUES (31003131, 'out for delivery', TO\_DATE('2024-06-03', 'YYYY-MM-DD'), 1, 10001);

INSERT INTO orders VALUES (32003232, 'delivered', TO\_DATE('2024-06-04', 'YYYY-MM-DD'), 2, 20002);

INSERT INTO orders VALUES (33003333, 'order placed', TO\_DATE('2024-06-05', 'YYYY-MM-DD'), 3, 30003);

INSERT INTO orders VALUES (34003434, 'shipped', TO\_DATE('2024-06-06', 'YYYY-MM-DD'), 4, 40004);

INSERT INTO orders VALUES (35003535, 'out for delivery', TO\_DATE('2024-06-07', 'YYYY-MM-DD'), 5, 50005);

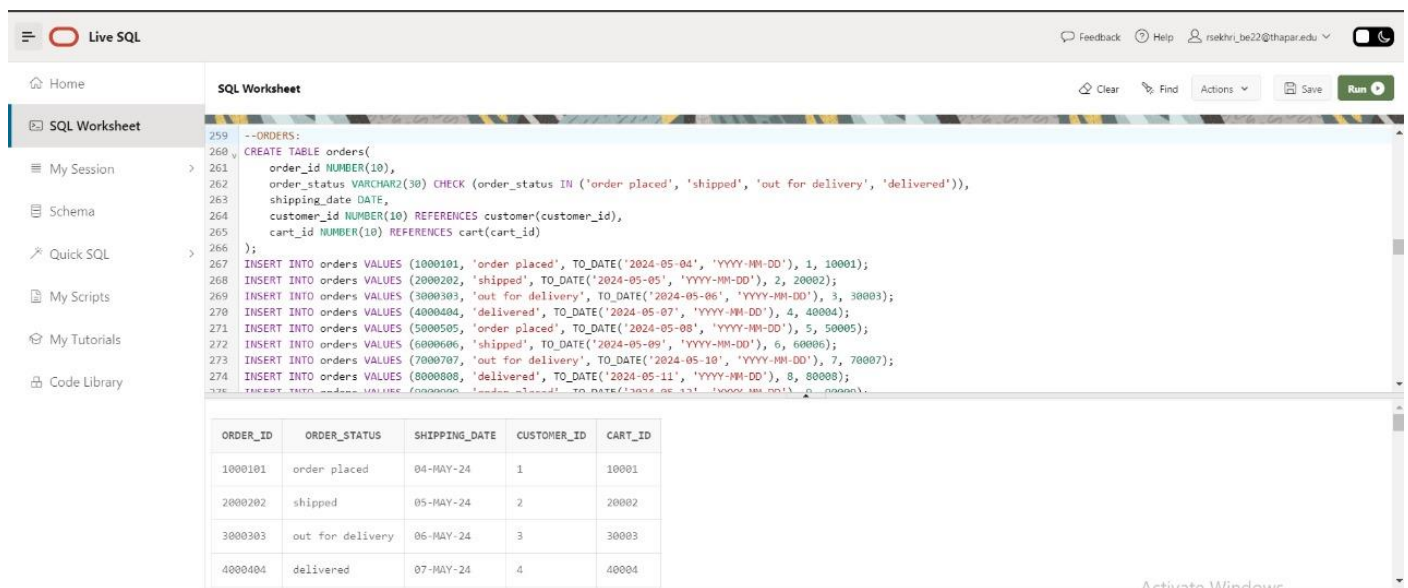
INSERT INTO orders VALUES (36003636, 'delivered', TO\_DATE('2024-06-08', 'YYYY-MM-DD'), 6, 60006);

INSERT INTO orders VALUES (37003737, 'order placed', TO\_DATE('2024-06-09', 'YYYY-MM-DD'), 7, 70007);

INSERT INTO orders VALUES (38003838, 'shipped', TO\_DATE('2024-06-10', 'YYYY-MM-DD'), 8, 80008);

INSERT INTO orders VALUES (39003939, 'out for delivery', TO\_DATE('2024-06-11', 'YYYY-MM-DD'), 9, 90009);

INSERT INTO orders VALUES (40004040, 'delivered', TO\_DATE('2024-06-12', 'YYYY-MM-DD'), 10, 10010);



The screenshot displays a web-based SQL editor titled "Live SQL". The main area shows a SQL script for creating an 'orders' table and inserting data. The script includes a CREATE TABLE statement with columns: order\_id (NUMBER(10)), order\_status (VARCHAR2(30) with a CHECK constraint), shipping\_date (DATE), customer\_id (NUMBER(10) with a REFERENCES constraint), and cart\_id (NUMBER(10) with a REFERENCES constraint). It follows with ten INSERT INTO statements, each adding a new order with a unique ID, status, date, customer ID, and cart ID.

ORDER_ID	ORDER_STATUS	SHIPPING_DATE	CUSTOMER_ID	CART_ID
1000101	order placed	04-MAY-24	1	10001
2000202	shipped	05-MAY-24	2	20002
3000303	out for delivery	06-MAY-24	3	30003
4000404	delivered	07-MAY-24	4	40004

## 8. USER LOGIN TABLE:

```
CREATE TABLE user_login(  
    email varchar2(30) unique,  
    password varchar2(16) not null,  
    customer_id references customer(customer_id)  
);
```

INSERT INTO user\_login VALUES ('rajesh.kumar@gmail.com', 'rajesh001', 1);

INSERT INTO user\_login VALUES ('priya.singh@gmail.com', 'priya002', 2);

INSERT INTO user\_login VALUES ('amit.patel@gmail.com', 'amit003', 3);

INSERT INTO user\_login VALUES ('divya.sharma@gmail.com', 'divya004', 4);

INSERT INTO user\_login VALUES ('sanjay.gupta@gmail.com', 'sanjay005', 5);

```

INSERT INTO user_login VALUES ('sneha.verma@gmail.com', 'sneha006', 6);
INSERT INTO user_login VALUES ('ritu.mehta@gmail.com', 'ritu007', 7);
INSERT INTO user_login VALUES ('anil.yadav@gmail.com', 'anil008', 8);
INSERT INTO user_login VALUES ('pooja.malhotra@gmail.com', 'pooja009', 9);
INSERT INTO user_login VALUES ('kiran.reddy@gmail.com', 'kiran010', 10);
INSERT INTO user_login VALUES ('manoj.tiwari@gmail.com', 'manoj011', 11);
INSERT INTO user_login VALUES ('neha.kapoor@gmail.com', 'neha012', 12);
INSERT INTO user_login VALUES ('vivek.joshi@gmail.com', 'vivek013', 13);
INSERT INTO user_login VALUES ('anjali.singh@gmail.com', 'anjali014', 14);
INSERT INTO user_login VALUES ('suresh.sharma@gmail.com', 'suresh015', 15);
INSERT INTO user_login VALUES ('deepika.patel@gmail.com', 'deepika016', 16);
INSERT INTO user_login VALUES ('rahul.gupta@gmail.com', 'rahul017', 17);
INSERT INTO user_login VALUES ('riya.malhotra@gmail.com', 'riya018', 18);
INSERT INTO user_login VALUES ('alok.singh@gmail.com', 'alok019', 19);
INSERT INTO user_login VALUES ('aditi.verma@gmail.com', 'aditi020', 20);
INSERT INTO user_login VALUES ('suman.yadav@gmail.com', 'suman021', 21);
INSERT INTO user_login VALUES ('ravi.kumar@gmail.com', 'ravi022', 22);
INSERT INTO user_login VALUES ('preeti.sharma@gmail.com', 'preeti023', 23);
INSERT INTO user_login VALUES ('akash.gupta@gmail.com', 'akash024', 24);
INSERT INTO user_login VALUES ('ananya.singh@gmail.com', 'ananya025', 25);
INSERT INTO user_login VALUES ('vikas.malhotra@gmail.com', 'vikas026', 26);
INSERT INTO user_login VALUES ('juhi.patel@gmail.com', 'juhi027', 27);
INSERT INTO user_login VALUES ('rajeshwari.singh@gmail.com', 'rajeshwari028', 28);
INSERT INTO user_login VALUES ('kunal.sharma@gmail.com', 'kunal029', 29);
INSERT INTO user_login VALUES ('shreya.gupta@gmail.com', 'shreya030', 30);

```

Live SQL

SQL Worksheet

```

309 --User Login:
310 create table user_login(
311     email varchar2(30) unique,
312     password varchar2(16) not null,
313     customer_id references customer(customer_id)
314 );
315 INSERT INTO user_login VALUES ('rajesh.kumar@gmail.com', 'rajesh001', 1);
316 INSERT INTO user_login VALUES ('priya.singh@gmail.com', 'priya002', 2);
317 INSERT INTO user_login VALUES ('amit.patel@gmail.com', 'amit003', 3);
318 INSERT INTO user_login VALUES ('divya.sharma@gmail.com', 'divya004', 4);
319 INSERT INTO user_login VALUES ('sanjay.gupta@gmail.com', 'sanjay005', 5);
320 INSERT INTO user_login VALUES ('sneha.verma@gmail.com', 'sneha006', 6);
321 INSERT INTO user_login VALUES ('ritu.mehta@gmail.com', 'ritu007', 7);
322 INSERT INTO user_login VALUES ('anil.yadav@gmail.com', 'anil008', 8);
323 INSERT INTO user_login VALUES ('pooja.mahotra@gmail.com', 'pooja009', 9);
324 INSERT INTO user_login VALUES ('kiran.reddy@gmail.com', 'kiran010', 10);
325 INSERT INTO user_login VALUES ('laxmi.kumar@gmail.com', 'laxmi011', 11);

```

EMAIL	PASSWORD	CUSTOMER_ID
rajesh.kumar@gmail.com	rajesh001	1
priya.singh@gmail.com	priya002	2
amit.patel@gmail.com	amit003	3
divya.sharma@gmail.com	divya004	4

Activate Windows

## 9. ORDER ITEM ID:

CREATE TABLE order\_item(

order\_item\_id NUMBER(12) PRIMARY KEY,

price NUMBER(10) NOT NULL,

quantity NUMBER(3) NOT NULL,

cart\_id NUMBER(10) REFERENCES cart(cart\_id)

);

INSERT INTO order\_item VALUES (900101, 1000, 2, 10001);

INSERT INTO order\_item VALUES (900102, 750, 3, 20002);

INSERT INTO order\_item VALUES (900201, 1200, 1, 30003);

INSERT INTO order\_item VALUES (900301, 500, 2, 40004);

INSERT INTO order\_item VALUES (900302, 800, 1, 50005);

INSERT INTO order\_item VALUES (900401, 1500, 1, 60006);

INSERT INTO order\_item VALUES (900501, 2000, 2, 70007);

INSERT INTO order\_item VALUES (900601, 600, 1, 80008);

INSERT INTO order\_item VALUES (900602, 900, 3, 90009);

INSERT INTO order\_item VALUES (900701, 300, 2, 10010);

INSERT INTO order\_item VALUES (900801, 400, 1, 11011);

INSERT INTO order\_item VALUES (900901, 1100, 1, 12012);

INSERT INTO order\_item VALUES (901001, 750, 2, 13013);

INSERT INTO order\_item VALUES (901002, 800, 1, 14014);

INSERT INTO order\_item VALUES (901101, 2000, 1, 15015);  
INSERT INTO order\_item VALUES (901201, 500, 3, 16016);  
INSERT INTO order\_item VALUES (901202, 900, 2, 17017);  
INSERT INTO order\_item VALUES (901301, 1200, 1, 18018);  
INSERT INTO order\_item VALUES (901401, 700, 2, 19019);  
INSERT INTO order\_item VALUES (901501, 1500, 1, 20020);  
INSERT INTO order\_item VALUES (901502, 800, 3, 21021);  
INSERT INTO order\_item VALUES (901601, 600, 1, 22022);  
INSERT INTO order\_item VALUES (901701, 500, 2, 23023);  
INSERT INTO order\_item VALUES (901801, 900, 1, 24024);  
INSERT INTO order\_item VALUES (901901, 2000, 1, 25025);  
INSERT INTO order\_item VALUES (901902, 1100, 2, 26001);  
INSERT INTO order\_item VALUES (902001, 750, 1, 27002);  
INSERT INTO order\_item VALUES (902101, 800, 2, 28003);  
INSERT INTO order\_item VALUES (902201, 1200, 1, 29004);  
INSERT INTO order\_item VALUES (902301, 700, 1, 30005);  
INSERT INTO order\_item VALUES (902401, 1500, 2, 22022);  
INSERT INTO order\_item VALUES (902501, 800, 1, 30005);  
INSERT INTO order\_item VALUES (902502, 600, 1, 25025);  
INSERT INTO order\_item VALUES (902601, 500, 3, 20002);  
INSERT INTO order\_item VALUES (902701, 900, 1, 40004);  
INSERT INTO order\_item VALUES (902801, 2000, 2, 70007);  
INSERT INTO order\_item VALUES (902901, 1100, 1, 21021);  
INSERT INTO order\_item VALUES (903001, 750, 1, 26001);  
INSERT INTO order\_item VALUES (903101, 800, 3, 17017);  
INSERT INTO order\_item VALUES (903201, 1200, 1, 19019);  
INSERT INTO order\_item VALUES (903301, 700, 1, 20020);  
INSERT INTO order\_item VALUES (903401, 1500, 2, 30003);  
INSERT INTO order\_item VALUES (903501, 800, 1, 50005);  
INSERT INTO order\_item VALUES (903601, 600, 1, 60006);  
INSERT INTO order\_item VALUES (903701, 500, 3, 17017);



The screenshot shows the Live SQL interface with an SQL Worksheet. The code in the worksheet is as follows:

```

346 --ORDER_ITEM_ID|
347 CREATE TABLE order_item(
348     order_item_id NUMBER(12) PRIMARY KEY,
349     price NUMBER(10) NOT NULL,
350     quantity NUMBER(3) NOT NULL,
351     cart_id NUMBER(10) REFERENCES cart(cart_id)
352 );
353 INSERT INTO order_item VALUES (900101, 1000, 2, 10001);
354 INSERT INTO order_item VALUES (900102, 750, 3, 20002);
355 INSERT INTO order_item VALUES (900201, 1200, 1, 30003);
356 INSERT INTO order_item VALUES (900301, 500, 2, 40004);
357 INSERT INTO order_item VALUES (900302, 800, 1, 50005);
358 INSERT INTO order_item VALUES (900401, 1500, 1, 60006);
359 INSERT INTO order_item VALUES (900501, 2000, 2, 70007);
360 INSERT INTO order_item VALUES (900601, 600, 1, 80008);
361 INSERT INTO order_item VALUES (900602, 900, 3, 90009);
362 INSERT INTO order_item VALUES (900701, 300, 2, 10010);

```

Below the code, a table preview shows the data inserted into the 'order\_item' table:

ORDER_ITEM_ID	PRICE	QUANTITY	CART_ID
900101	1000	2	10001
900102	750	3	20002
900201	1200	1	30003
900301	500	2	40004

## 10. WISHLIST TABLE :

CREATE TABLE wishlist(

wishlist\_id number(10) primary key,

customer\_id references customer(customer\_id),

product\_id references product(product\_id)

);

INSERT INTO wishlist VALUES (111, 1, 5001);

INSERT INTO wishlist VALUES (112, 2, 5003);

INSERT INTO wishlist VALUES (113, 3, 5005);

INSERT INTO wishlist VALUES (114, 4, 5007);

INSERT INTO wishlist VALUES (115, 5, 5009);

INSERT INTO wishlist VALUES (116, 6, 5011);

INSERT INTO wishlist VALUES (117, 7, 5013);

INSERT INTO wishlist VALUES (118, 8, 5015);

INSERT INTO wishlist VALUES (119, 9, 5017);

INSERT INTO wishlist VALUES (120, 10, 5019);

INSERT INTO wishlist VALUES (121, 11, 5021);

INSERT INTO wishlist VALUES (122, 12, 5023);

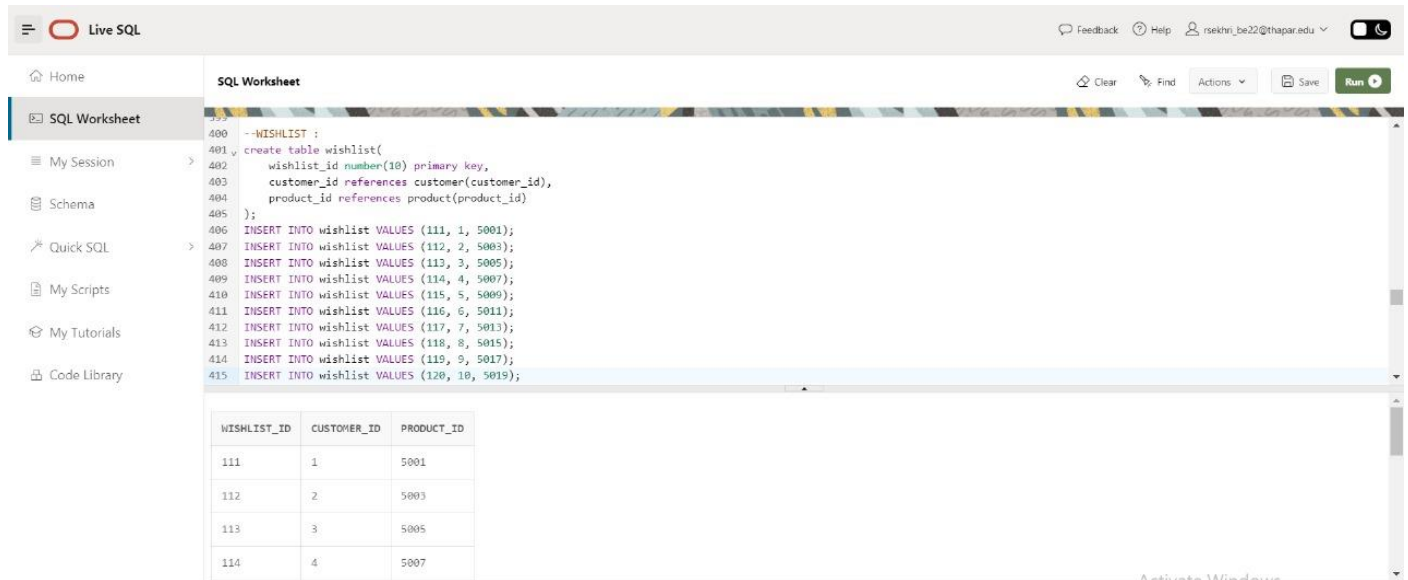
INSERT INTO wishlist VALUES (123, 13, 5025);

INSERT INTO wishlist VALUES (124, 14, 5027);

```

INSERT INTO wishlist VALUES (125, 15, 5029);
INSERT INTO wishlist VALUES (126, 16, 5031);
INSERT INTO wishlist VALUES (127, 17, 5033);
INSERT INTO wishlist VALUES (128, 18, 5035);
INSERT INTO wishlist VALUES (129, 19, 5037);
INSERT INTO wishlist VALUES (130, 20, 5039);

```



## 11. PAYMENT TABLE:

```

CREATE TABLE payment(
    payment_id number(10) primary key,
    methods varchar2(15) check (methods in('UPI', 'credit card','debit card','cash on delivery','wallet')),
    customer_id references customer(customer_id),
    product_id references product(product_id),
    order_item_id references order_item(order_item_id),
    payment_date date
);

```

```

INSERT INTO payment VALUES (1563, 'UPI', 1, 5001, 900101, TO_DATE('2023-09-05', 'YYYY-MM-DD'));

```

```

INSERT INTO payment VALUES (1564, 'credit card', 2, 5002, 903601, TO_DATE('2023-09-08', 'YYYY-MM-DD'));

```

```

INSERT INTO payment VALUES (1565, 'debit card', 3, 5003, 900102, TO_DATE('2023-09-12', 'YYYY-MM-DD'));

```

```

INSERT INTO payment VALUES (1566, 'cash on delivery', 4, 5004, 900201, TO_DATE('2023-09-15', 'YYYY-MM-DD'));

```

INSERT INTO payment VALUES (1567, 'wallet', 5, 5005, 900301, TO\_DATE('2023-09-20', 'YYYY-MM-DD'));

INSERT INTO payment VALUES (1568, 'UPI', 6, 5006, 900302, TO\_DATE('2023-09-25', 'YYYY-MM-DD'));

INSERT INTO payment VALUES (1569, 'credit card', 7, 5007, 900401, TO\_DATE('2023-09-28', 'YYYY-MM-DD'));

INSERT INTO payment VALUES (1570, 'debit card', 8, 5008, 900501, TO\_DATE('2023-10-02', 'YYYY-MM-DD'));

INSERT INTO payment VALUES (1571, 'cash on delivery', 9, 5009, 900601, TO\_DATE('2023-10-05', 'YYYY-MM-DD'));

INSERT INTO payment VALUES (1572, 'wallet', 10, 5010, 900901, TO\_DATE('2023-10-08', 'YYYY-MM-DD'));

INSERT INTO payment VALUES (1573, 'UPI', 11, 5011, 901001, TO\_DATE('2023-10-12', 'YYYY-MM-DD'));

INSERT INTO payment VALUES (1574, 'credit card', 12, 5012, 901002, TO\_DATE('2023-10-15', 'YYYY-MM-DD'));

INSERT INTO payment VALUES (1575, 'debit card', 13, 5013, 901101, TO\_DATE('2023-10-18', 'YYYY-MM-DD'));

INSERT INTO payment VALUES (1576, 'cash on delivery', 14, 5014, 901201, TO\_DATE('2023-10-22', 'YYYY-MM-DD'));

INSERT INTO payment VALUES (1577, 'wallet', 15, 5015, 901202, TO\_DATE('2023-10-25', 'YYYY-MM-DD'));

The screenshot shows a web-based SQL editor interface. The top bar includes a 'Live SQL' logo, user information, and navigation icons. The left sidebar contains a 'SQL Worksheet' tab and a list of items: Home, My Session, Schema, Quick SQL, My Scripts, My Tutorials, and Code Library. The main area displays SQL code for creating a 'payment' table and inserting 15 rows of data. Below the code, a table preview shows the first five rows of the data.

```
427 --PAYMENT :
428 create table payment(
429     payment_id number(10) primary key,
430     methods varchar2(15) check (methods in('UPI', 'credit card','debit card','cash on delivery','wallet')),
431     customer_id references customer(customer_id),
432     product_id references product(product_id),
433     order_item_id references order_item(order_item_id),
434     payment_date date
435 );
436 INSERT INTO payment VALUES (1563, 'UPI', 1, 5001, 900101, TO_DATE('2023-09-05', 'YYYY-MM-DD'));
437 INSERT INTO payment VALUES (1564, 'credit card', 2, 5002, 903601, TO_DATE('2023-09-08', 'YYYY-MM-DD'));
438 INSERT INTO payment VALUES (1565, 'debit card', 3, 5003, 900102, TO_DATE('2023-09-12', 'YYYY-MM-DD'));
439 INSERT INTO payment VALUES (1566, 'cash on delivery', 4, 5004, 900201, TO_DATE('2023-09-15', 'YYYY-MM-DD'));
440 INSERT INTO payment VALUES (1567, 'wallet', 5, 5005, 900301, TO_DATE('2023-09-20', 'YYYY-MM-DD'));
441 INSERT INTO payment VALUES (1568, 'UPI', 6, 5006, 900302, TO_DATE('2023-09-25', 'YYYY-MM-DD'));
442 INSERT INTO payment VALUES (1569, 'credit card', 7, 5007, 900401, TO_DATE('2023-09-28', 'YYYY-MM-DD'));
443 INSERT INTO payment VALUES (1570, 'debit card', 8, 5008, 900501, TO_DATE('2023-10-02', 'YYYY-MM-DD'));
```

PAYMENT_ID	METHODS	CUSTOMER_ID	PRODUCT_ID	ORDER_ITEM_ID	PAYMENT_DATE
1563	UPI	1	5001	900101	05-SEP-23
1564	credit card	2	5002	903601	08-SEP-23
1565	debit card	3	5003	900102	12-SEP-23
1567	wallet	5	5005	900301	20-SEP-23

# PL/SQL QUERIES

## 1. PROCEDURE:

Procedure which returns the total quantity of product with the given ID

CODE:

```
CREATE OR REPLACE PROCEDURE PROD_DETAILS(p_id in varchar)
```

```
is
```

```
  quan number(2);
```

```
  BEGIN
```

```
    select quantity into quan from product where product_id=p_id;
```

```
    dbms_output.put_line('product quantity is :'|| quan);
```

```
  exception
```

```
    when no_data_found then
```

```
      dbms_output.put_line('Sorry no such product exist !!');
```

```
  end;
```

```
DECLARE
```

```
BEGIN
```

```
  prod_details(5001);
```

```
end;
```

```
498 v create or replace procedure prod_details(p_id in varchar)
499 is
500   quan number(2);
501 v begin
502     select quantity into quan from product where product_id=p_id;
503     dbms_output.put_line('product quantity is :'|| quan);
504 v exception
505     when no_data_found then
506       dbms_output.put_line('Sorry no such product exist !!');
507 end;
508
509 v declare
510 begin
511   prod_details(5001);
512 end;|
```

```
Statement processed.
product quantity is :50
```

## 2. FUNCTION:

Function which returns total number of products which a particular seller sells

### CODE:

```
CREATE OR REPLACE FUNCTION TOTALPRODUCTS(sId in varchar)
```

```
return number
```

```
is
```

```
total number(2):=0;
```

```
begin
```

```
select count(*) into total
```

```
from product
```

```
where seller_id=sId;
```

```
return total;
```

```
end;
```

```
DECLARE
```

```
total_products NUMBER;
```

```
BEGIN
```

```
total_products := totalProducts('1012');
```

```
DBMS_OUTPUT.PUT_LINE('Total products: ' || total_products);
```

```
END;
```

```
481 v create or replace function totalProducts(sId in varchar)
482     return number is
483     total number(2):=0;
484 v     begin
485     select count(*) into total
486     from product
487     where seller_id=sId;
488     return total;
489     end;
490 v DECLARE
491     total_products NUMBER;
492 v BEGIN
493     total_products := totalProducts('1012');
494     DBMS_OUTPUT.PUT_LINE('Total products: ' || total_products);
495 END;|
```

Statement processed.

Total products: 2

### 3. CURSOR:

Procedure which returns the brand of product with the cost less than the given cost

CODE:

```
CREATE OR REPLACE PROCEDURE cost_filter(
  c IN NUMBER,
  H_and_M IN VARCHAR2
)
IS
  cs product.price%TYPE;
  b product.brand%TYPE;
  id product.product_id%TYPE;
  CURSOR cf IS
    SELECT product_id, price, brand FROM product WHERE price < c AND brand = H_and_M;
BEGIN
  OPEN cf;
  LOOP
    FETCH cf INTO id, cs, b;
    EXIT WHEN cf%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE('Product ' || id || ' has cost ' || cs || ' and the brand is ' || b);
  END LOOP;
  CLOSE cf;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Sorry, no such products exist');
END;
DECLARE
  -- Declare other variables if needed
BEGIN
  -- Call the procedure with the desired value for c and H_and_M
  cost_filter(1000, 'H&M');
```

END;

```
481 v CREATE OR REPLACE PROCEDURE cost_filter(c IN NUMBER,H_and_M IN VARCHAR2)IS
482   cs product.price%TYPE;
483   b product.brand%TYPE;
484   id product.product_id%TYPE;
485 v   CURSOR cf IS
486     SELECT product_id, price, brand FROM product WHERE price < c AND brand = H_and_M;
487 v BEGIN
488   OPEN cf;
489 v   LOOP
490     FETCH cf INTO id, cs, b;
491     EXIT WHEN cf%NOTFOUND;
492     DBMS_OUTPUT.PUT_LINE('Product ' || id || ' has cost ' || cs || ' and the brand is ' || b);
493   END LOOP;
494   CLOSE cf;
495 v EXCEPTION
496   WHEN NO_DATA_FOUND THEN
497     DBMS_OUTPUT.PUT_LINE('Sorry, no such products exist');
498 END;
499 v DECLARE
```

```
Statement processed.
Product 5008 has cost 599 and the brand is H&M
Product 5024 has cost 499 and the brand is H&M
```

#### 4. TRIGGER:

##### CODE:

```
CREATE OR REPLACE FUNCTION total_cost(cId IN VARCHAR)
RETURN NUMBER
IS
    total NUMBER := 0;
BEGIN
    SELECT SUM(product.price)
    INTO total
    FROM product, cart
    WHERE product.product_id = cart.product_id
    AND cart.cart_id = cId;

    RETURN total;
END;
CREATE OR REPLACE TRIGGER before_pay_up
BEFORE INSERT ON payment
FOR EACH ROW
DECLARE
    total_cost_val NUMBER;
BEGIN
    total_cost_val := total_cost(:new.cart_id);
    :new.total_amount := total_cost_val;
END;
INSERT INTO payment (payment_id, methods, customer_id, product_id, order_item_id, payment_date)
VALUES (1575, 'debit card', 13, 5013, 901101, '18-OCT-23');
```

SQL Worksheet

```
512 ✓ | CREATE OR REPLACE FUNCTION total_cost(cId IN VARCHAR)
513     RETURN NUMBER
514 IS
515     total NUMBER := 0;
516 ✓ BEGIN
517     SELECT SUM(product.price)
518     INTO total
519     FROM product, cart
520     WHERE product.product_id = cart.product_id
521     AND cart.cart_id = cId;
522
523     RETURN total;
524 END;
525 ✓ CREATE OR REPLACE TRIGGER before_pay_up
526 BEFORE INSERT ON payment
527 FOR EACH ROW
528 DECLARE
529     total_cost_val NUMBER;
530 ✓ BEGIN
531     total_cost_val := total_cost(:new.cart_id);
532     :new.total_amount := total_cost_val;
533 END;
```

PAYMENT_ID	METHODS	CUSTOMER_ID	PRODUCT_ID	ORDER_ITEM_ID	PAYMENT_DATE
1575	debit card	13	5013	901101	18-OCT-23



# SQL QUERIES

## 1. Query 1:

```
select product_id,product_name, color,price, seller_id
from product
where (brand='levis' and sizes='L' and gender='female' and quantity>0);
```

The screenshot shows a web-based SQL editor interface. At the top, there's a header with 'Feedback', 'Help', and a user profile 'rsekhi\_be22@thapar.edu'. Below the header, the 'SQL Worksheet' tab is active. The editor contains the following SQL code:

```
565 END;
566
567
568 -- SQL QUERIES
569 -- CUSTOMERS WHO HAVE PURCHASED NOTHING
570 Select * from customer where customer_id not in (select customer_id from Payment);
571
572 select product_id,product_name, color,price, seller_id from product where (brand='levis' and sizes='L' and gender='female' and quantity>0);
573
```

Below the code editor, the results of the query are displayed in a table:

PRODUCT_ID	PRODUCT_NAME	COLOR	PRICE	SELLER_ID
5002	womens jeans	blue	1499	1002

A 'Download CSV' button is located below the table.

## 2. Query 2:

```
SELECT
    p.product_id, p.product_name, p.price,
    p.commission, p.gender, p.sizes,
    p.color, p.brand, p.stock,
    c.cart_id, c.customer_id
FROM
    product p
JOIN
    cart c ON p.product_id = c.product_id;
```

The screenshot shows a web-based SQL editor interface titled 'Live SQL'. On the left, there's a sidebar with navigation links: 'Home', 'SQL Worksheet', 'My Session', 'Schema', 'Quick SQL', 'My Scripts', 'My Tutorials', and 'Code Library'. The 'SQL Worksheet' tab is active. The editor contains the following SQL code:

```
573 SELECT
574     p.product_id, p.product_name, p.price,
575     p.commission, p.gender, p.sizes,
576     p.color, p.brand, p.stock,
577     c.cart_id, c.customer_id
578 FROM
579     product p
580 JOIN
581     cart c ON p.product_id = c.product_id;
```

Below the code editor, the results of the query are displayed in a table:

PRODUCT_ID	PRODUCT_NAME	PRICE	COMMISSION	GENDER	SIZES	COLOR	BRAND	STOCK	CART_ID	CUSTOMER_ID
5001	mens t-shirt	999	150	male	M	black	adidas	100	10001	1
5002	womens jeans	1499	200	female	L	blue	levis	80	20002	2
5003	mens polo shirt	899	120	male	XL	green	puma	120	30003	3
5003	mens polo shirt	899	120	male	XL	green	puma	120	25001	26
5004	womens dress	1999	250	female	M	red	zara	60	40004	4
5005	mens sneakers	2999	300	male	L	white	nike	50	50005	5
5006	womens sandals	799	100	female	S	brown	bata	90	60006	6
5007	mens formal shirt	1299	180	male	XXL	blue	van heusen	70	70007	7

A 'Activate Windows' watermark is visible in the bottom right corner of the results table.



## **CONCLUSION**

The development of the e-commerce management system project utilizing DBMS and SQL has been instrumental in creating a dynamic platform for online shopping websites. This project has successfully addressed key functionalities essential for efficient e-commerce operations through meticulous database design, implementation, and integration.

A primary achievement of this project lies in creating a well-structured database schema that accommodates diverse entities such as customers, products, orders, payments, and cart by adhering to normalization. This project has successfully addressed key functionalities essential for efficient e-commerce operations through meticulous database design, implementation, and integration principles, the database ensures data integrity, minimizes redundancy, and optimizes query performance, thus laying a robust foundation for the system's scalability and reliability.

Moreover, the incorporation of SQL queries, triggers, and stored procedures has significantly enhanced the system's functionality and efficiency. SQL queries enable seamless retrieval, manipulation, and analysis of data, thereby empowering users to navigate the platform effortlessly. Triggers and stored procedures automate routine tasks, enforce business rules, and maintain data consistency, thereby streamlining various aspects of the e-commerce workflow.

Additionally, continuous optimization of database performance, security, and scalability will be crucial to meet the evolving demands of the online retail landscape.

In conclusion, the e-commerce management system project represents a significant milestone in leveraging DBMS and SQL technologies to create a sophisticated platform for online shopping websites.

## **REFERENCES**

1. <https://github.com/arlotfi79/E-commerce-Management-System>
2. <https://www.myntra.com/>
3. [https://www2.hm.com/en\\_in/index.html](https://www2.hm.com/en_in/index.html)
4. <https://www.nykaafashion.com/>
5. <https://www.zara.com/in/>
6. <https://www.pantaloons.com/>
7. <https://github.com/arlotfi79/E-commerce-Management-System>