```sql
CREATE DATABASE customer_shopping_data;
USE customer_shopping_data;

----- import DATASET
SELECT * FROM customer_shopping_data;


--------------------------------------
-- 1. Remove Duplicates
-- 2. Standarize the Data
-- 3. Null Values or blank  values
-- 4. Remove Any Columns

----------------------------------------
-- 1. Remove Duplicates

CREATE TABLE customer_shopping LIKE customer_shopping_data;

SELECT * FROM customer_shopping;

INSERT customer_shopping SELECT * FROM customer_shopping_data;

SELECT *,
ROW_NUMBER() OVER(
PARTITION BY invoice_no, customer_id,gender,age,category,quantity,price,payment_method,
invoice_date,shopping_mall) AS ROW_NUM
FROM customer_shopping;

SELECT * FROM customer_shopping;
SELECT * FROM customer_shopping_2;

CREATE TABLE `customer_shopping_2` (
  `invoice_no` text,
  `customer_id` text,
  `gender` text,
  `age` int DEFAULT NULL,
  `category` text,
  `quantity` int DEFAULT NULL,
  `price` double DEFAULT NULL,
  `payment_method` text,
  `invoice_date` text,
  `shopping_mall` text,
  `row_num` INT
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


INSERT INTO  customer_shopping_2
SELECT *,
ROW_NUMBER() OVER(
PARTITION BY invoice_no, customer_id,gender,age,category,quantity,price,payment_method,
invoice_date,shopping_mall) AS ROW_NUM
FROM customer_shopping;

-- Disable safe updates temporarily
SET SQL_SAFE_UPDATES = 0;

-- Delete duplicate rows based on row_num
DELETE FROM customer_shopping_2
WHERE row_num > 1;

-- Re-enable safe updates
SET SQL_SAFE_UPDATES = 1;

-- View final cleaned data
SELECT * FROM customer_shopping_2;


--------------------------
-- 2. Standarizing Data
```

```sql
SELECT * FROM customer_shopping_2;

SELECT customer_id, TRIM(customer_id)
FROM customer_shopping_2;

UPDATE customer_shopping_2
SET  customer_id = TRIM(customer_id);

SELECT payment_method, TRIM(payment_method) FROM customer_shopping_2;

SELECT DISTINCT invoice_no
FROM customer_shopping_2
ORDER BY 1;


SELECT `invoice_date`,
STR_TO_DATE(`invoice_date`,'%d/%m/%Y')
FROM customer_shopping_2;

UPDATE customer_shopping_2
SET invoice_date = STR_TO_DATE(invoice_date,'%d/%m/%Y');


-- UPDATE  price  decimal TO ROUND NUMBER.

UPDATE customer_shopping_2
SET price = ROUND(price, 0);


ALTER TABLE customer_shopping_2
MODIFY COLUMN `invoice_date` DATE;

----------------------------------------
-- 3. Null Values or blank  values of Eevery Columns

SELECT * FROM customer_shopping_2;

SELECT DISTINCT category
FROM customer_shopping_2
WHERE category IS NULL
OR category = '';

SELECT DISTINCT customer_id
FROM customer_shopping_2
WHERE customer_id IS NULL
OR customer_id = '';

SELECT DISTINCT invoice_no
FROM customer_shopping_2
WHERE invoice_no IS NULL
OR invoice_no = '';

SELECT DISTINCT age
FROM customer_shopping_2
WHERE age IS NULL
OR age = '';

SELECT DISTINCT payment_method
FROM customer_shopping_2
WHERE payment_method IS NULL
OR payment_method = '';

SELECT DISTINCT price
FROM customer_shopping_2
WHERE price IS NULL
OR price = '';

SELECT DISTINCT invoice_date
FROM customer_shopping_2
```

```sql
WHERE invoice_date IS NULL
OR invoice_date = '';

SELECT DISTINCT shopping_mall
FROM customer_shopping_2
WHERE shopping_mall IS NULL
OR shopping_mall = '';

SELECT DISTINCT shopping_mall
FROM customer_shopping_2
WHERE shopping_mall IS NULL
OR shopping_mall = '';


-- CHecking the Datatypes
DESC customer_shopping_2;
---

ALTER TABLE customer_shopping_2
DROP COLUMN  row_num;

SELECT * FROM customer_shopping_2;

-- EDA DATA ANALYSIS AND DATA PRE-PROCESSING

SELECT MAX(price)
FROM customer_shopping_2;

SELECT MAX(price)
FROM customer_shopping_2
WHERE price = 5250
ORDER BY customer_id DESC ;

----------------------
-- Total Price According customer_id

SELECT customer_id, ROUND(SUM(price),0) AS SUM_OF_PRICE
FROM customer_shopping_2
GROUP BY customer_id
ORDER BY 2 DESC;

----------------------
-- Total Price According MIN & MAX DATE

SELECT MIN(`invoice_date`), MAX(`invoice_date`), SUM(price)
FROM customer_shopping_2;

----------------------
-- Total Price According category

SELECT category, ROUND(SUM(price),0)
FROM customer_shopping_2
GROUP BY category
ORDER BY 2 DESC;

----------------------
-- Total Price According shopping_mall

SELECT shopping_mall, ROUND(SUM(price),0)
FROM customer_shopping_2
GROUP BY  shopping_mall
ORDER BY 2 DESC;

----------------------
-- Total  Price According DISTINCT YEAR

SELECT YEAR(`invoice_date`), ROUND(SUM(price),0)
FROM customer_shopping_2
GROUP BY  YEAR(`invoice_date`)
```

```sql
ORDER BY 1 DESC;
----------------------
-- Total  Price According the MONTH with Date

SELECT SUBSTRING(`invoice_date`, 6,2) AS 'MONTH',SUBSTRING(`invoice_date`, 9, 2) as dat,
ROUND(SUM(price),0)
FROM customer_shopping_2
WHERE SUBSTRING(`invoice_date`, 6,2) IS NOT NULL
GROUP BY  SUBSTRING(`invoice_date`, 6,2), SUBSTRING(`invoice_date`, 9, 2)
ORDER BY 1 ASC;

----------------------
-- Total  Price According the MONTH WITH YEAR

SELECT SUBSTRING(`invoice_date`,1,7) AS 'MONTH', ROUND(SUM(price),0)
FROM customer_shopping_2
WHERE SUBSTRING(`invoice_date`, 1,7) IS NOT NULL
GROUP BY  SUBSTRING(`invoice_date`, 1,7)
ORDER BY 1 ASC;

-- Total  Price According the shopping_mall  WITH YEAR

SELECT DISTINCT shopping_mall, YEAR(`invoice_date`), ROUND(SUM(price),0) AS SUM_OF_PRICE
FROM customer_shopping_2
GROUP BY shopping_mall, YEAR(`invoice_date`)
ORDER BY 3 DESC;


SELECT * FROM customer_shopping_2;
```