```
create database Garments_shop;

use Garments_shop;

select * from garments;

describe garments;

ALTER TABLE garments
CHANGE COLUMN `Unit Price (?)` `Unit Price (₹)` DECIMAL(10,2),
CHANGE COLUMN `Total Amount (?)` `Total Amount (₹)` DECIMAL(10,2);

describe garments;

UPDATE garments
SET `Date` = NULL
WHERE `Date` = '';

SET SQL_SAFE_UPDATES = 0;


ALTER TABLE garments
modify COLUMN `Date` DATE;

 -- adding the date values in the null --
select * from garments;
```

-- we found that the null values are present in the Date column where the invoice no at INV028, INV064 and INV090 --

-- we can add 5th March and 6th march for the following missing dates --

```
UPDATE garments
SET `Date` = '2024-03-05'
WHERE `Invoice No` = 'INV028';

update garments
set `Date` = '2024-03-06' where `Invoice No` in ('INV064', 'INV090');

select * from garments where `Invoice No` in ('INV028', 'INV064', 'INV090');
```

-- finding the missing values in Category column --

```
select * from garments
where `Category` = '';
```

-- filling up the missing values in category column as per the product name --

```
UPDATE garments
set `Category` = 'Women'
where `Product Name` = 'Kurti';

update garments
set `Category` = 'Kids'
where `Product Name` in ('Frock', 'Kids Jeans', 'Kids T-Shirt');

update garments
set `Category` = 'Men'
where `Product Name` in ('Formal Shirt', 'Jeans');
```

```
select * from garments;


 -- Next to fill up the details in Size column --

select  * from garments

where `Size`  = '';


-- we found that several values in saree are having the sizes such as 28, XXL which are invalid, so update everything with "Free" --

update garments

set `Size` = 'Free'

where `Product Name` = 'Saree';


select  * from garments

where `Product Name` = 'Saree';

-- found that details have been updaed correctly  --


-- Let's normalize the size and its numbers. where shirts, kids t shirts, frocks and kurti's numbers to e changed as size such as M, S, XL

 -- and Jeans, kids jeans and leggings where size changed to numbers.


UPDATE garments

SET `Size` = CASE

   WHEN `Product Name` IN ('Casual Shirt', 'Formal Shirt') THEN

      CASE

         WHEN `Size` = '28' THEN 'S'

         WHEN `Size` IN ('30', '32') THEN 'M'

         WHEN `Size` = '34' THEN 'L'

         WHEN `Size` IS NULL OR `Size` = '' THEN 'L'
```

```
        ELSE `Size` -- Keep original value if none of the above
      END
    ELSE `Size` -- Don't change for other product types
END
WHERE `Product Name` IN ('Casual Shirt', 'Formal Shirt');

select * from garments
where `Product Name` IN ('Casual Shirt', 'Formal Shirt');

-- found that details updated successfully. --

-- Lets change the same for Frock  as below --
UPDATE garments
SET `Size` = CASE
    WHEN `Product Name` = 'Frock' THEN
      CASE
        WHEN `Size` = '30' THEN 'S'
        WHEN `Size` = '32' THEN 'M'
        WHEN `Size` IN ('34', 'XXL', 'L') THEN 'Free'
        ELSE `Size` -- Keep original value if none of the above
      END
    ELSE `Size` -- Don't change for other product types
END
WHERE `Product Name` = 'Frock';

select * from garments
where `Product Name` = 'Frock';
```

-- details updated successfully --

-- lets update the details for Kids T-shirt and Kurti --

```
UPDATE garments
SET `Size` = CASE
  WHEN `Product Name` = 'Kids T-Shirt' THEN
    CASE
      WHEN `Size` = '28' THEN 'S'
      WHEN `Size` IN ('30', '32') THEN 'M'
      WHEN `Size` = '34' THEN 'L'
      WHEN `Size` IS NULL OR `Size` = '' THEN 'M'
      ELSE `Size`
    END
  WHEN `Product Name` = 'Kurti' THEN
    CASE
      WHEN `Size` IN ('28', '30') THEN 'S'
      WHEN `Size` = '34' THEN 'M'
      ELSE `Size`
    END
  ELSE `Size`
END
WHERE `Product Name` IN ('Kids T-Shirt', 'Kurti');

-- Before update
SELECT `Invoice No`, `Product Name`, `Size`
FROM garments
WHERE `Product Name` IN ('Kids T-Shirt', 'Kurti');
```

-- details updated successfully --

-- lets change the details for Jeans, Kids Jeans and Leggings --

```
UPDATE garments
SET `Size` = CASE
  WHEN `Product Name` = 'Jeans' THEN
    CASE
      WHEN `Size` = 'S' THEN '28'
      WHEN `Size` = 'M' THEN '30'
      WHEN `Size` = 'L' THEN '34'
      WHEN `Size` = 'XL' THEN '38'
      WHEN `Size` = 'XXL' THEN '40'
      WHEN `Size` IS NULL OR `Size` = '' THEN '34'
      ELSE `Size`
    END
  WHEN `Product Name` = 'Kids Jeans' THEN
    CASE
      WHEN `Size` = 'S' THEN '28'
      WHEN `Size` = 'M' THEN '30'
      WHEN `Size` = 'XL' THEN '34'
      WHEN `Size` = 'XXL' THEN '36'
      WHEN `Size` IS NULL OR `Size` = '' THEN '30'
      ELSE `Size`
    END
  WHEN `Product Name` = 'Leggings' THEN
    CASE
```

```
        WHEN `Size` = 'XL' THEN '32'
        WHEN `Size` = 'L' THEN '30'
        WHEN `Size` = 'S' THEN '28'
        WHEN `Size` = 'XXL' THEN '34'
        ELSE `Size`
      END
    ELSE `Size`
END
WHERE `Product Name` IN ('Jeans', 'Kids Jeans', 'Leggings');

SELECT `Invoice No`, `Product Name`, `Size`
FROM garments
WHERE `Product Name` IN ('Jeans', 'Kids Jeans', 'Leggings')
ORDER BY `Product Name`, `Size`;

 -- details are updated successfully --
 select * from garments
 where `Payment Mode` = '';

 -- found that several details are mising in payment mode. Lets fill --
UPDATE garments
SET `Payment Mode` = 'Cash'
WHERE `Payment Mode` IS NULL OR `Payment Mode` = '';

-- Now we have the cleaned dataset where missing values are filled, few columns got
normalized. --

SELECT `Product Name`,
```

```
    SUM(Quantity) AS Total_Quantity,
    SUM(`Total Amount (₹)`) AS Total_Sales
FROM garments
GROUP BY `Product Name`
ORDER BY Total_Sales DESC
LIMIT 5;

-- 3) Which sizes are most frequently sold per category?
SELECT
  Category,
  Size,
  SUM(Quantity) AS Total_Quantity_Sold
FROM
  garments
GROUP BY
  Category, Size
ORDER BY
  Category ASC, Total_Quantity_Sold DESC;



SELECT
  Category,
  Size,
  Total_Quantity_Sold,
  dense_rank() OVER (PARTITION BY Category ORDER BY Total_Quantity_Sold DESC) AS
Size_Rank
FROM (
  SELECT
```

```
    Category,
    Size,
    SUM(Quantity) AS Total_Quantity_Sold
  FROM garments
  GROUP BY Category, Size
) AS size_summary
ORDER BY Category, Size_Rank;



SELECT
  `Date`,
  `Product Name`,
  ROUND(AVG(`Unit Price (₹)`), 2) AS Avg_Unit_Price,
  ROUND(AVG(Quantity), 2) AS Avg_Quantity
FROM garments
GROUP BY `Date`, `Product Name`
ORDER BY `Date`, `Product Name`;

SELECT
  `Date`,
  SUM(`Total Amount (₹)`) AS Total_Revenue
FROM garments
GROUP BY `Date`
HAVING Total_Revenue > 10000
ORDER BY Total_Revenue DESC
limit 10;

SELECT
```

```
    `Product Name`,
    ROUND(AVG(`Unit Price (₹)`), 2) AS Avg_Unit_Price,
    SUM(Quantity) AS Total_Quantity_Sold
FROM garments
GROUP BY `Product Name`
ORDER BY Avg_Unit_Price DESC, Total_Quantity_Sold ASC;



SELECT
    `Product Name`,
    COUNT(DISTINCT Date) AS Days_Sold,
    SUM(Quantity) AS Total_Quantity_Sold,
    ROUND(AVG(`Unit Price (₹)`), 2) AS Avg_Unit_Price
FROM garments
GROUP BY `Product Name`
HAVING Days_Sold > 5 AND Total_Quantity_Sold > 10
ORDER BY Total_Quantity_Sold DESC;



SELECT
    `Product Name`,
    COUNT(DISTINCT CONCAT(YEAR(Date), '-', MONTH(Date))) AS Months_Sold,
    SUM(Quantity) AS Total_Quantity_Sold,
    ROUND(AVG(`Unit Price (₹)`), 2) AS Avg_Unit_Price,
    SUM(`Total Amount (₹)`) AS Total_Revenue
FROM garments
GROUP BY `Product Name`
HAVING Total_Quantity_Sold < 60 AND Months_Sold < 3
```

```
ORDER BY Total_Quantity_Sold ASC, Total_Revenue ASC;



SELECT
  CASE
    WHEN `Total Amount (₹)` < 1000 THEN 'Low(<1000)'
    WHEN `Total Amount (₹)` BETWEEN 1000 AND 5000 THEN 'Medium(btn
1000&5000)'
    ELSE 'High(>5000)'
  END AS Price_Segment,
  `Payment Mode`,
  COUNT(*) AS Transactions
FROM garments
GROUP BY Price_Segment, `Payment Mode`
ORDER BY Price_Segment, Transactions DESC;

SELECT
  Category,
  ROUND(AVG(`Total Amount (₹)`), 2) AS Avg_Bill_Per_Transaction
FROM garments
GROUP BY Category
ORDER BY Avg_Bill_Per_Transaction DESC;
```