

Table of Contents:

[View Statistics](#)

[Get Holiday List](#)

[Add Holiday](#)

[Maintain Population](#)

[View Product by Category Report](#)

[View Actual vs. Predicted Revenue for Couches and Sofas Report](#)

[Get Available State List](#)

[View Store Revenue by Year by State Report](#)

[View Groundhog Day Outdoor Furniture Report](#)

[Get Year and Month List](#)

[View State with Highest Volume by Category Report](#)

[View Revenue by Population Report](#)

[View Childcare Sales Volume Report](#)

[View Restaurant Impact on Category Sales Report](#)

[View Advertising Campaign Analysis Report](#)

View Statistics

Abstract Code¹

- Show ***“Holiday Maintenance”, “View Product by Category Report”, “View Actual vs. Predicted Revenue for Couches and Sofas Report”, “View Store Revenue by Year by State Report”, “View Groundhog Day Outdoor Furniture Report”, “View State with Highest Volume by Category Report”, “View Revenue by Population Report”, “View Childcare Sales Volume Report”, “View Restaurant Impact on Category Sales Report”, “View Advertising Campaign Analysis Report”, and “Population Maintenance”*** buttons/links on the **Dashboard** form.
- Upon:
 - Click ***Holiday Maintenance*** button – Jump to the **Holiday Maintenance** form.
 - Click ***View Product by Category Report*** button – Jump to the **View Product by Category Report** task.
 - Click ***View Actual vs. Predicted Revenue for Couches and Sofas Report*** button – Jump to the **View Actual vs. Predicted Revenue for Couches and Sofas Report** task.
 - Click ***View Store Revenue by Year by State Report*** button – Jump to the **Get Available State List** task.
 - Click ***View Groundhog Day Outdoor Furniture Report*** button – Jump to the **View Groundhog Day Outdoor Furniture Report** task.
 - Click ***View State with Highest Volume by Category Report*** button – Jump to the **Get Year and Month List** task.
 - Click ***View Revenue by Population Report*** button – Jump to the **View Revenue by Population Report** task.
 - Click ***View Childcare Sales Volume Report*** button – Jump to the **View Childcare Sales Volume Report** task.
 - Click ***View Restaurant Impact on Category Sales Report*** button – Jump to the **View Restaurant Impact on Category Sales Report** task.
 - Click ***View Advertising Campaign Analysis Report*** button – Jump to the **View Advertising Campaign Analysis Report** task.
 - Click ***Population Maintenance*** button – Jump to the **Population Maintenance** form.
- Display statistics for “the count of stores”, “count of stores offering food (have a restaurant, a snack bar, or both)”, “count of stores offering childcare”, “count of products”, and “count of distinct advertising campaigns” on the **Dashboard** form.
 - Show “the count of stores”.
 - Query for total count of Store_Number in the **STORE** table.
 - Display the total count.

```
SELECT COUNT(Store_Number) FROM STORE;
```

¹ MySQL is applied to our submission.

- Show “count of stores offering food”.
 - Query for the total count of Store_Number in the [STORE](#) table that has either or both Has_Restaurant and Has_Snack_Bar value as true.
 - Display the total count.

```
SELECT COUNT(Store_Number)
FROM STORE
WHERE Has_Restaurant IS TRUE OR Has_Snack_Bar IS TRUE;
```

- Show “count of stores offering childcare”.
 - Query for the total count of Store_Number in the [STORE](#) table that has a Childcare center association in the [CHILDCARE](#) table.
 - Display the total count.

```
SELECT COUNT(Store_Number)
FROM STORE
WHERE Time_Limit IS NOT NULL;
```

- Show “count of products”.
 - Query for the total count of PID in the [PRODUCT](#) table.
 - Display the total count.

```
SELECT COUNT(PID) FROM PRODUCT;
```

- Show “count of distinct advertising campaigns”.
 - Query for the total count of Description in [ADVERTISING_CAMPAIGN](#) table.
 - Display the total count.

```
SELECT COUNT(Description) FROM ADVERTISING\_CAMPAIGN;
```

Get Holiday List

Abstract Code

- User clicked on the **Holiday Maintenance** button from the **Dashboard** form.
- Run the **Get Holiday List** task: query for information about the available Name field from the [HOLIDAY](#) table.
 - Display the holiday name list.

```
SELECT `Date`, `Name` FROM HOLIDAY;
```

- Upon:
 - User enters Holiday Name (\$HolidayName) in input textbox and selects Date (\$HolidayDate) in Calendar Dropdown.
 - Click **Add Holiday** button –
 - Jump to the **Add Holiday** task.
- When ready, user can click on the **Return** button to return to the **Dashboard** form.

Add Holiday

Abstract Code

- User enters Holiday Name (\$HolidayName) in input textbox and selects Date (\$HolidayDate) in Calendar Dropdown.
- Click **Add Holiday** button.
- Run the **Add Holiday** task:
 - If data validation passed for both holiday name and date in Client Side, then:
 - If holiday name does not exist but date exists:
 - Store the holiday name in **HOLIDAY** table and link its date with **DAY** table.
 - Go back to **Holiday Maintenance** form and show success message.
 - If both holiday name and date do not exist:
 - Store the holiday name and date in both **HOLIDAY** and **DAY** tables.
 - Go back to **Holiday Maintenance** form and show success message.

```
INSERT INTO `DAY` (`Date`)
SELECT '$HolidayDate'
FROM `DAY`
WHERE NOT EXISTS (
    SELECT 1 FROM `DAY` WHERE `Date` = '$HolidayDate')
LIMIT 1;

INSERT INTO HOLIDAY (`Date`, `Name`)
SELECT '$HolidayDate', '$HolidayName'
FROM HOLIDAY
WHERE NOT EXISTS (
    SELECT 1 FROM HOLIDAY WHERE `Date` = '$HolidayDate' AND
    `Name` = '$HolidayName')
LIMIT 1;
```

- Else: display the invalid error message in **Holiday Maintenance** form.
- When ready, user can click on the **Return** button to return to the **Dashboard** form.

Maintain Population

Abstract Code

- User clicked on the **Population Maintenance** button from the **Dashboard** form.
- Run the **Get City List** task: query for information about the available *State_Location*, *City_Name*, and *Population* fields from the **CITY** table.

- Display *State_Location* list in ascending order in the drop-down list.

```
SELECT DISTINCT State_Location  
FROM CITY  
ORDER BY State_Location ASC;
```

- Display the available *City_Name* list in ascending order based on the selected *\$State_Location* in the drop-down list.

```
SELECT City_Name  
FROM CITY  
WHERE State_Location = '$State_Location'  
ORDER BY City_Name ASC;
```

- Display *Population* based on the selected *\$State_Location* and *\$City_Name* in the view population textbox.

```
SELECT Population  
FROM CITY  
WHERE State_Location = '$State_Location' AND City_Name = '$City_Name';
```

- User edits population textbox.
- Upon:
 - Click **Update Population** button –
 - Jump to the **Update Population** task.
- Run the **Update Population** task.
 - If data validation passed for *\$Population* in Client Side, then:
 - If the updated population entered is the same as the original population, do nothing.
 - Else: the updated population entered is different from the original population, update the *Population* in the **CITY** table.

```
UPDATE CITY  
SET Population = '$Population'  
WHERE State_Location = '$State_Location' AND City_Name = '$City_Name'  
AND Population <> '$Population';
```

- Else: display the invalid error message in **Population Maintenance** form.
- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Product by Category Report

Abstract Code

- User clicked on the ***View Product by Category Report*** button from the **Dashboard** form.
- Run the ***View Product by Category Report*** task: query for each category from **CATEGORY**, **ASSIGNED**, and **PRODUCT** tables, including those without products.
 - Find all Category_Name data (from the **CATEGORY** table).
 - For each Category_Name including those without products:
 - Find total number of products by counting their PID data (from the **PRODUCT** table).
 - Find minimum, average, and maximum Retail_Price data for all products (from the **PRODUCT** table).
 - Sort by Category_Name in ascending order.

```
SELECT C.Category_Name, COUNT(P.PID) AS Cnt_Product,  
MIN(P.Retail_Price) AS Min_RtlPrc, AVG(P.Retail_Price) AS Avg_RtlPrc,  
MAX(P.Retail_Price) AS Max_RtlPrc  
FROM CATEGORY AS C  
LEFT JOIN ASSIGNED AS A ON C.Category_Name = A.Category_Name  
LEFT JOIN PRODUCT AS P ON A.PID = P.PID  
GROUP BY C.Category_Name  
ORDER BY C.Category_Name ASC;
```

- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Actual vs. Predicted Revenue for Couches and Sofas Report

Abstract Code

- User clicked on the ***View Actual vs. Predicted Revenue for Couches and Sofas Report*** button from the **Dashboard** form.
- Run the **View Actual vs. Predicted Revenue for Couches and Sofas Report** task:
 - Find the 'Couches and Sofas' category (from the **CATEGORY** table).
 - For each product in 'Couches and Sofas' category:
 - Find PID, Product_Name, and Retail_Price data (from the **PRODUCT** table).
 - Find the total number of units ever sold by aggregating Quantity (from the **SALE** table).
 - Find the total number of units sold at a discount by aggregating Quantity (from the **SALE** table) when the product has a Discount_Price (from the **DISCOUNT** table).
 - Find the total number of units sold at retail price by aggregating Quantity (from the **SALE** table) when the product doesn't have a Discount_Price (from the **DISCOUNT** table).
 - Find the actual revenue by aggregating Total_Amount (from the **SALE** table).
 - Total_Amount (from the **SALE** table) is calculated based on the Date, Quantity, and individual item price.
 - Individual item price equals Retail_Price (from the **PRODUCT** table) when the product doesn't have a Discount_Price. Otherwise, individual item price equals Discount_Price (from the **DISCOUNT** table).
 - Find the predicted revenue by aggregating the following multiplication result: Retail_Price (from the **PRODUCT** table) * Quantity (from the **SALE** table) * quantity multiplier. The quantity multiplier equals 0.75 when the product has a Discount_Price (from the **DISCOUNT** table), otherwise, the quantity multiplier equals 1.
 - Find the revenue difference by subtracting predicted revenue from actual revenue.
 - If the revenue difference is greater than \$5000 (positive or negative): Display revenue difference and sort in descending order.

```
SELECT P.PID, P.Product_Name, P.Retail_Price,
SUM(IFNULL(S.Quantity,0)) AS Tot_UnitSold,
SUM(IF(D.Discount_Price IS NULL,0,1) * IFNULL(S.Quantity,0)) AS Tot_UnitSold_AtDsc,
SUM(IF(D.Discount_Price IS NULL,1,0) * IFNULL(S.Quantity,0)) AS Tot_UnitSold_AtRtl,
SUM(IFNULL(S.Total_Amount,0)) AS Act_Revenue,
SUM(P.Retail_Price * IFNULL(S.Quantity,0) * IF(D.Discount_Price IS NULL, 1, 0.75)) AS
Pred_Revenue,
(SUM(IFNULL(S.Total_Amount,0)) - SUM(P.Retail_Price * IFNULL(S.Quantity,0) *
IF(D.Discount_Price IS NULL, 1, 0.75))) AS Diff_Act_Pred_Revenue
FROM CATEGORY AS C
LEFT JOIN ASSIGNED AS A ON C.Category_Name = A.Category_Name
LEFT JOIN PRODUCT AS P ON A.PID = P.PID
LEFT JOIN SALE AS S ON P.PID = S.PID
LEFT JOIN DISCOUNT AS D ON S.Date = D.Date AND S.PID = D.PID
WHERE C.Category_Name = 'Couches and Sofas'
GROUP BY P.PID
HAVING Diff_Act_Pred_Revenue > 5000 OR Diff_Act_Pred_Revenue < -5000
ORDER BY Diff_Act_Pred_Revenue DESC;
```

- When ready, user can click on the **Return** button to return to the **Dashboard** form.

Get Available State List

Abstract Code

- User clicked on **View Store Revenue by Year by State Report** button from the **Dashboard** form.
- Run the **Get Available State List** task: query for information about the available *State_Location* field from the **CITY** table.

- Display *State_Location* list in ascending order on the drop-down list.

```
SELECT DISTINCT State_Location
FROM CITY
ORDER BY State_Location ASC;
```

- On the drop-down list, show **Run Report** button.
- Upon:
 - Click **Run Report** button –
 - If *\$State_Location* is selected – Jump to the **View State Revenue by Year by State Report** task.
 - If *\$State_Location* is empty – Display a message asking for user input.
- When ready, user can click on the **Return** button to return to the **Dashboard** form.

View Store Revenue by Year by State Report

Abstract Code

- User clicked on the **Run Report** button from the drop-down list.
- Run the **View State Revenue by Year by State Report** task:
 - Find all stores from the **STORE** table based on the selected **\$State_Location**.
 - For each store and each year,
 - Display Store_Number, Street_Address, and City_Name (from the **STORE** table), and the year of Date (from the **SALE** table).
 - Find the total revenue by aggregating all products' sale revenue based on Total_Amount (from the **SALE** table).
 - Sort by year ascendingly, and then sort by total revenue descendingly.

```
SELECT STORE.Store_Number, Street_Address, City_Name, YEAR(Date),  
SUM(IFNULL(Total_Amount, 0)) AS Revenue  
FROM STORE  
LEFT JOIN SALE ON STORE.Store_Number = SALE.Store_Number  
WHERE State_Location = '$State_Location'  
GROUP BY STORE.Store_Number, YEAR(Date)  
ORDER BY YEAR(Date) ASC, Revenue DESC;
```

- When ready, user can click on the **Return** button to return to the **Dashboard** form.

View Groundhog Day Outdoor Furniture Report

Abstract Code

- User clicked on the **View Groundhog Day Outdoor Furniture Report** button from the **Dashboard** form.
- Run the **View Groundhog Day Outdoor Furniture Report** task:
 - Get the outdoor furniture category (from the **CATEGORY** table).
 - For each year:
 - Find year based on Date (from the **SALE** table).
 - Find total number of products sold by aggregating Quantity (from the **SALE** table) in that year.
 - Find average number of products sold per day by dividing total number of products sold by 365.
 - Find total number of products sold on Groundhog Day by aggregating Quantity (from the **SALE** table) on Feb 2 in that year.

- Sort by year in ascending order.

```
SELECT YEAR(Date),  
SUM(IFNULL(Quantity, 0)) AS Tot_Quantity,  
(SUM(IFNULL(Quantity, 0)) / 365) AS Avg_Quantity,  
SUM(IF(MONTH(Date)=2 AND DAY(Date)=2,1,0) * IFNULL(Quantity, 0)) AS  
GhDay_Quantity  
FROM CATEGORY AS C  
LEFT JOIN ASSIGNED AS A ON C.Category_Name = A.Category_Name  
LEFT JOIN PRODUCT AS P ON A.PID = P.PID  
LEFT JOIN SALE AS S ON P.PID = S.PID  
WHERE C.Category_Name = 'Outdoor Furniture'  
GROUP BY YEAR(Date)  
ORDER BY YEAR(Date) ASC;
```

- When ready, user can click on the **Return** button to return to the **Dashboard** form.

Get Year and Month List

Abstract Code

- User clicked on the **View State with Highest Volume by Category Report** button from the **Dashboard** form.
- Run the **Get Year and Month List** task: query for information about the available *Year* and *Month* fields from the **DAY** table.
 - Display *Year* list in descending order in the drop-down list.

```
SELECT DISTINCT YEAR(`Date`) AS `Year`  
FROM `DAY`  
ORDER BY `Year` DESC;
```

- Display *Month* list in descending order based on the selected **\$Year** in the drop-down list.

```
SELECT DISTINCT MONTH(`Date`) AS `Month`  
FROM `DAY`  
WHERE YEAR(`Date`) = '$Year'  
ORDER BY `Month` DESC;
```

- On the drop-down list, show **Run Report** button.
- Upon:

- Click **Run Report** button –
 - If both \$Year and \$Month are selected – Jump to the **View State with Highest Volume by Category Report** task.
 - If one or both fields are empty – Display a message asking for user input.
- When ready, user can click on the **Return** button to return to the **Dashboard** form.

View State with Highest Volume by Category Report

Abstract Code

- User clicked on the **Run Report** button from the drop-down list.
- If data validation is successful for both \$Year and \$Month input fields, then proceed.
- Run the **View State with Highest Volume by Category Report** task:
 - Based on the selected \$Year and \$Month,
 - For each category,
 - Find the states that sold the highest number of units in the category.
 - If two or more states tied for having the highest number of units, then save all those states.
 - Display Category_Name (from the CATEGORY table), State_Location (from the STORE table), and the number of units that were sold by stores in that state by aggregating Quantity (from the SALE table).
 - Sort by Category_Name ascendingly.

```

SELECT C.Category_Name, T.State_Location, SUM(IFNULL(S.Quantity,0)) AS Tot_UnitSold
FROM CATEGORY AS C
LEFT JOIN ASSIGNED AS A ON C.Category_Name = A.Category_Name
LEFT JOIN PRODUCT AS P ON A.PID = P.PID
LEFT JOIN SALE AS S ON P.PID = S.PID AND YEAR(S.`Date`) = '$Year' AND
MONTH(S.`Date`) = '$Month'
LEFT JOIN STORE AS T ON S.Store_Number = T.Store_Number
GROUP BY C.Category_Name, T.State_Location
HAVING Tot_UnitSold >= ALL
(
    SELECT SUM(IFNULL(S2.Quantity,0))
    FROM CATEGORY AS C2
    LEFT JOIN ASSIGNED AS A2 ON C2.Category_Name = A2.Category_Name
    LEFT JOIN PRODUCT AS P2 ON A2.PID = P2.PID
    LEFT JOIN SALE AS S2 ON P2.PID = S2.PID AND YEAR(S2.`Date`) = '$Year' AND
    MONTH(S2.`Date`) = '$Month'
    LEFT JOIN STORE AS T2 ON S2.Store_Number = T2.Store_Number
    WHERE C2.Category_Name = C.Category_Name
    GROUP BY C2.Category_Name, T2.State_Location
)
ORDER BY C.Category_Name ASC;

```

- When ready, user can click on the **Return** button to return to the **Dashboard** form.

View Revenue by Population Report

Abstract Code

- User clicked on the **View Revenue by Population Report** button from the **Dashboard** form.
- Run the **View Revenue by Population Report** task:
 - Find the Date, Population, and Total_Amount from the **SALE**, **STORE**, and **CITY** tables.
 - For each year of Date:
 - Find total revenue for each city size category by aggregating Total_Amount.
 - Display total revenue based on year and city size category:
 - Row: Years, sorted in ascending order.
 - Column: City size categories, sorted in ascending order. Specifically, SmallCity, MediumCity, LargeCity, and ExtraLargeCity.

```

SELECT
    YEAR(SRC.`Date`) AS Years,
    SUM(IF(Population < 3700000, Total_Amount, 0)) AS SmallCity,
    SUM(IF(Population >= 3700000 AND Population < 6700000,
    Total_Amount, 0)) AS MediumCity,
    SUM(IF(Population >= 6700000 AND Population < 9000000,
    Total_Amount, 0)) AS LargeCity,
    SUM(IF(Population >= 9000000, Total_Amount, 0)) AS
    ExtraLargeCity
FROM (
    SELECT
        SALE.`Date`,
        SALE.Total_Amount,
        CITY.Population
    FROM CITY
        INNER JOIN STORE ON CITY.State_Location =
STORE.State_Location AND CITY.City_Name = STORE.City_Name
        INNER JOIN SALE ON STORE.Store_Number =
SALE.Store_Number
    ) AS SRC
GROUP BY YEAR(SRC.`Date`)
ORDER BY YEAR(SRC.`Date`) ASC;
    
```

- When ready, user can click on the **Return** button to return to the **Dashboard** form.

View Childcare Sales Volume Report

Abstract Code

- User clicked on the **View Childcare Sales Volume Report** button from the **Dashboard** form.
- Run the **View Childcare Sales Volume Report** task:
 - Find distinct Childcare_Category based on Time_Limit (from the **STORE** table). If Childcare_Category doesn't have a value, set Childcare_Category to 'No childcare'.
 - For each Sale_Year_Month based on `Date` (from the **SALE** table) in last 12 months:
 - Find the total sales by aggregating Total_Amount (from the **SALE** table) in each Childcare_Category.
 - Display total sales in a tabular form:
 - Row: Sale_Year_Month, sorted in ascending order.
 - Column: Childcare_Category.

```

SET @Sql = "";

SELECT @Sql:=CONCAT(@Sql,
'SUM(IF(Childcare_Category=\',Childcare_Category,\',',Total_Amount,0)) AS \',
Childcare_Category,\',')
FROM (
    SELECT DISTINCT
    IF(STORE.Time_Limit IS NOT NULL, CAST(STORE.Time_Limit AS CHAR(10)),
    "No childcare") AS Childcare_Category
    FROM STORE
) AS TL;

SET @Sql = CONCAT('SELECT Sale_Year_Month,',LEFT(@Sql, LENGTH(@Sql)-1),'
FROM (
    SELECT
    DATE_FORMAT(SALE.`Date`, "%Y-%m") AS Sale_Year_Month,
    SALE.Total_Amount AS Total_Amount,
    IF( STORE.Time_Limit IS NOT NULL, CAST(STORE.Time_Limit AS CHAR(10)),
    "No childcare") AS Childcare_Category
    FROM STORE
    INNER JOIN SALE ON STORE.Store_Number = SALE.Store_Number
    WHERE SALE.`Date` >= DATE_ADD( LAST_DAY(DATE_SUB(CURDATE(),
    INTERVAL 12 MONTH) ), INTERVAL 1 DAY)
) AS REV
GROUP BY Sale_Year_Month
ORDER BY Sale_Year_Month ASC');

PREPARE stmt FROM @Sql;
EXECUTE stmt;

```

- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Restaurant Impact on Category Sales Report

Abstract Code

- User clicked on the ***View Restaurant Impact on Category Sales Report*** button from the **Dashboard** form.
- Run the **View Restaurant Impact on Category Sales Report** task:
 - Get Category_Name that has product(s) assigned (from the **CATEGORY**, **ASSIGNED**, and **PRODUCT** tables) and has sales data (from the **SALE** table).
 - Get `Store Type` based on Has_Restaurant (from the **STORE** table). If Has_Restaurant is true, set `Store Type` to 'Restaurant', otherwise, set `Store Type` to 'Non-restaurant'.
 - For each Category_Name and `Store Type`:
 - Include either 'Restaurant' or 'Non-restaurant', even if there's no sales data.
 - Find `Quantity Sold` by aggregating all products' Quantity (from the **SALE** table).
 - Sort by Category_Name ascendingly and then by `Store Type` with 'Non-restaurant' listed first.

```

SELECT COLS.Category, COLS.`Store Type`, IFNULL(ACT.`Quantity Sold`, 0)
FROM
  ( ( SELECT C01.Category_Name AS Category, 'Restaurant' AS `Store Type`
    FROM CATEGORY AS C01)
  UNION
  ( SELECT C02.Category_Name AS Category, 'Non-restaurant' AS `Store Type`
    FROM CATEGORY AS C02)
  ) AS COLS
LEFT JOIN
  (SELECT
    C.Category_Name AS Category,
    IF(T.Has_Restaurant IS TRUE, 'Restaurant', 'Non-restaurant') AS `Store Type`,
    SUM(S.Quantity) AS `Quantity Sold`
  FROM CATEGORY AS C
  INNER JOIN ASSIGNED AS A ON C.Category_Name = A.Category_Name
  INNER JOIN PRODUCT AS P ON A.PID = P.PID
  INNER JOIN SALE AS S ON P.PID = S.PID
  INNER JOIN STORE AS T ON S.Store_Number = T.Store_Number
  GROUP BY Category, `Store Type`
  ) AS ACT
ON COLS.Category = ACT.Category AND COLS.`Store Type` = ACT.`Store Type`
ORDER BY COLS.Category ASC, COLS.`Store Type` ASC;

```

- When ready, user can click on the ***Return*** button to return to the **Dashboard** form.

View Advertising Campaign Analysis Report

Abstract Code

- User clicked on the ***View Advertising Campaign Analysis Report*** button from the **Dashboard** form.
- Run the **View Advertising Campaign Analysis Report** task: query for information while a product has Discount_Price (from the [DISCOUNT](#) table).
 - For each product:
 - Display PID and Product_Name (from the [PRODUCT](#) table).
 - Find total quantity sold during campaign by aggregating Quantity (from the [SALE](#) table) in all discount sale days when any advertising campaign was active.
 - Find total quantity sold outside campaign by aggregating Quantity (from the [SALE](#) table) in all discount sale days when no advertising campaign was active.
 - Find the difference by subtracting quantity sold outside campaign from quantity sold during campaign.
 - Sort by difference in descending order and only display the top 10 followed by the bottom 10.


```

SELECT `Product ID`, `Product Name`, `Sold During Campaign`, `Sold Outside Campaign`, Difference
FROM(
  (
    SELECT P.PID AS `Product ID`, P.Product_Name AS `Product Name`,
    SUM(IF(A.`Description` IS NULL,0,1) * IFNULL(S.Quantity,0)) AS `Sold During Campaign`,
    SUM(IF(A.`Description` IS NULL,1,0) * IFNULL(S.Quantity,0)) AS `Sold Outside Campaign`,
    (SUM(IF(A.`Description` IS NULL,0,1) * IFNULL(S.Quantity,0)) -
    SUM(IF(A.`Description` IS NULL,1,0) * IFNULL(S.Quantity,0))) AS Difference
    FROM PRODUCT AS P
    INNER JOIN DISCOUNT AS I ON P.PID = I.PID
    LEFT JOIN SALE AS S ON I.`Date` = S.`Date` AND I.PID = S.PID
    LEFT JOIN DAY AS D ON S.`Date` = D.`Date`
    LEFT JOIN HOLD AS H ON D.`Date` = H.`Date`
    LEFT JOIN ADVERTISING_CAMPAIGN AS A ON H.`Description` = A.`Description`
    GROUP BY `Product ID`
    ORDER BY Difference DESC
    LIMIT 10
  )
  UNION
  (
    SELECT P2.PID AS `Product ID`, P2.Product_Name AS `Product Name`,
    SUM(IF(A2.`Description` IS NULL,0,1) * IFNULL(S2.Quantity,0)) AS `Sold During Campaign`,
    SUM(IF(A2.`Description` IS NULL,1,0) * IFNULL(S2.Quantity,0)) AS `Sold Outside Campaign`,
    (SUM(IF(A2.`Description` IS NULL,0,1) * IFNULL(S2.Quantity,0)) -
    SUM(IF(A2.`Description` IS NULL,1,0) * IFNULL(S2.Quantity,0))) AS Difference
    FROM PRODUCT AS P2
    INNER JOIN DISCOUNT AS I2 ON P2.PID = I2.PID
    LEFT JOIN SALE AS S2 ON I2.`Date` = S2.`Date` AND I2.PID = S2.PID
    LEFT JOIN DAY AS D2 ON S2.`Date` = D2.`Date`
    LEFT JOIN HOLD AS H2 ON D2.`Date` = H2.`Date`
    LEFT JOIN ADVERTISING_CAMPAIGN AS A2 ON H2.`Description` = A2.`Description`
    GROUP BY `Product ID`
    ORDER BY Difference ASC
    LIMIT 10
  )
) AS UN
ORDER BY Difference DESC;

```

- When ready, user can click on the **Return** button to return to the **Dashboard** form.