**[Table of Contents:](#Table_of_Contents)**

[View Statistics](#View_Statistics)

[Get Holiday List](#Get_Holiday_List)

[Add Holiday](#Add_Holiday)

[Maintain Population](#Maintain_Population)

[View Product by Category Report](#View_Product_by_Category_Report)

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

[Get Available State List](#Get_Available_State_List)

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

[View Groundhog Day Outdoor Furniture Report](#View_Groundhog_Day_Outdoor_Furniture)

[Get Year and Month List](#Get_Year_and_Month_List)

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

[View Revenue by Population Report](#View_Revenue_by_Population_Report)

[View Childcare Sales Volume Report](#View_Childcare_Sales_Volume_Report)

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

[View Advertising Campaign Analysis Report](#View_Advertising_Campaign_Analysis)

View Statistics

Abstract Code

* 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;

* + 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.

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;

* Go back to **Holiday Maintenance** form and show success message.
  + 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 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\_AtDsct,

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\_Locationis selected – Jump to the **View State Revenue by Year by State Report** task.
    - If $State\_Locationis 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 OUTER 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 and return the year (from the DAY table).
* For each year:
  + - Get the outdoor furniture category (from the CATEGORY table).
    - Get Quantity data for the products sold (from the SALE table) at the specific Date that year (from the DAY table); Find total number of products sold by aggregating Quantity in all sale days 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 (Feb 2) using Quantity and Date (from SALE and DAY tables).
  + Sort by year in ascending order.

SELECT D.Year, S.Quantity

FROM CATEGORY AS C, DAY AS D

INNER JOIN SALE AS S ON S.Date = D.Date

SUM(IFNULL(S.Quantity, 0)) AS Tot\_Quantity

(SUM(IFNULL(S.Quantity, 0)) / 365) AS Ave\_Quantity

WHERE C.Category\_Name = 'Outdoor Furniture'

(SELECT Quantity AS Grou\_Day\_Quantity

FROM SALE

WHERE C.Category\_Name = 'Outdoor Furniture' AND D.Date = Feb 2)

GROUP BY D.Year

ORDER BY D.Year 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 both *year* and *month* lists in descending order on the drop-down list.

SELECT YEAR(Date) AS Year, MONTH(Date) AS Month

FROM `DAY`

ORDER BY Year DESC, 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:
  + Find all Sales data from the SALE table based on the *year* and *month* (from the DAY table) selected.
  + Get all categories by using Category\_Name data (from the CATEGORY table).
  + In each category, aggregate each state’s sales Quantity data (from the SALE table) by adding up all stores (from the STORE table)’s sales Quantity in each state (from the CITY table).
  + Find the states that sold the highest number of units in each category.
    - If two or more states tied for having the greatest number of units, then save all those states and their total units.
  + Sort by category name ascendingly, display each category name, its corresponding states with the highest units sold and total units sold.

SELECT ASSIGNED.Category\_Name, STORE.State\_Location, MAX(SALE.Quantity) AS Max\_Quantity

FROM SALE INNER JOIN ASSIGNED ON SALE.PID = ASSIGNED.PID

INNER JOIN STORE ON STORE.Store\_Number = SALE.Store\_Number

WHERE YEAR(SALE.Date) = '$Year’ AND MONTH(SALE.Date) = '$Month’

GROUP BY ASSIGNED.Category\_Name, STORE.State\_Location

HAVING COUNT(\*) > 0

ORDER BY ASSIGNED.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 Year, City\_Size and sale’s Total\_Amount from the DAY, SALE, STORE, and CITY tables.
  + Group by year and City\_Size, and then aggregate the Total\_Amount as total revenue.
  + Display Total Revenue in a tabular form:
    - Row: year, sorted in ascending order.
    - Column: City\_Size, sorted in ascending order.

SELECT

SRC.Sale\_Year,

SRC.City\_Size,

SUM(SRC.Total\_Amount) AS Year\_Revenue

FROM (

SELECT

CASE CITY.Population

WHEN CITY.Population < 3700000 THEN "small"

WHEN (CITY.Population >= 3700000 AND CITY.Population < 6700000) THEN "medium"

WHEN (CITY.Population >= 6700000 AND CITY.Population < 9000000) THEN "large"

ELSE "extralarge" END AS City\_Size,

YEAR(SALE.Date) AS Sale\_Year,

SALE.Total\_Amount

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

INNER JOIN DAY ON SALE.Date = DAY.Date

) SRC

GROUP BY SRC.Sale\_Year, SRC.City\_Size

ORDER BY SRC.Sale\_Year ASC, FIELD(SRC.City\_Size, "small", "medium", "large", "extralarge") ;

* 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 and aggregate the store monthly sales from the SALE and STORE tables based on the Store\_Number (from the STORE table) and month (Date from the DAY and SALE tables) based on the last 12 months.
  + Find stores with and without childcare category from the STORE and CHILDCARE tables.
  + Group the total sales by month and by childcare category (no childcare offer will be grouped as category “No childcare”).
  + Display Total Sale in a tabular form:
    - Row: month, sorted in ascending order.
    - Column: childcare category.

SELECT

SRC.Store\_Number,

SRC. Childcare\_Category,

SUM(IF(Sale\_Month = 'Jan', Revenue, 0)) AS 'Jan',

SUM(IF(Sale\_Month = 'Feb', Revenue, 0)) AS 'Feb',

SUM(IF(Sale\_Month = 'Mar', Revenue, 0)) AS 'Mar',

SUM(IF(Sale\_Month = 'Apr', Revenue, 0)) AS 'Apr',

SUM(IF(Sale\_Month = 'May', Revenue, 0)) AS 'May',

SUM(IF(Sale\_Month = 'Jun', Revenue, 0)) AS 'Jun',

SUM(IF(Sale\_Month = 'Jul', Revenue, 0)) AS 'Jul',

SUM(IF(Sale\_Month = 'Aug', Revenue, 0)) AS 'Aug',

SUM(IF(Sale\_Month = 'Sep', Revenue, 0)) AS 'Sep',

SUM(IF(Sale\_Month = 'Oct', Revenue, 0)) AS 'Oct',

SUM(IF(Sale\_Month = 'Nov', Revenue, 0)) AS 'Nov',

SUM(IF(Sale\_Month = 'Dec', Revenue, 0)) AS 'Dec'

FROM

(

SELECT

STORE.Store\_Number,

DATE\_FORMAT(SALE.Date, "%b") AS Sale\_Month,

SALE.Total\_Amount AS Revenue,

IF(CHILDCARE.Time\_Limit IS NOT NULL, CAST(CHILDCARE.Time\_Limit AS CHAR(10)), "No Childcare") AS Childcare\_Category

FROM STORE

INNER JOIN SALE ON STORE.Store\_Number = SALE.Store\_Number AND SALE.Date >= Date\_add(Now(),interval - 12 month)

LEFT JOIN CHILDCARE ON STORE.Time\_Limit = CHILDCARE.Time\_Limit

) SRC

GROUP BY SRC.Store\_Number, SRC.Childcare\_Category

ORDER BY SRC.Store\_Number, SRC. Childcare\_Category ASC;

SELECT

SRC.Store\_Number,

SRC. Childcare\_Category,

SRC. Sale\_Month,

SUM(IFNULL(Revenue, 0)) AS Revenue

FROM

(

SELECT

STORE.Store\_Number,

DATE\_FORMAT(SALE.Date , '%M') AS Sale\_Month,

SALE.Total\_Amount AS Revenue,

IF(CHILDCARE.Time\_Limit IS NOT NULL, CAST(CHILDCARE.Time\_Limit AS CHAR(10)), "No Childcare") AS Childcare\_Category

FROM STORE

INNER JOIN SALE ON STORE.Store\_Number = SALE.Store\_Number AND SALE.Date >= Date\_add(Now(),interval - 12 month)

LEFT JOIN CHILDCARE ON STORE.Time\_Limit = CHILDCARE.Time\_Limit

) SRC

GROUP BY SRC.Store\_Number, SRC.Childcare\_Category

ORDER BY SRC.Store\_Number, SRC. Childcare\_Category, SRC. Sale\_Month ASC;

* 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 all Category\_Name data that has product(s) assigned (from CATEGORY and PRODUCT tables).
  + For each category:
    - Get products in this category (from CATEGORY and PRODUCT tables).
    - Find stores with Has\_Restaurant (from the STORE table) = TRUE:
      * Display store type with value “Restaurant”.
      * Find total quantity sold by aggregating all Quantity (from the SALE table) of all products for these stores.
    - Find stores with Has\_Restaurant (from the STORE table) = FALSE:
      * Display the store type with value “Non-Restaurant”.
      * Find total quantity sold by aggregating all Quantity (from the SALE table) of all products for these stores.
* Group by Category\_Name ascendingly and with “Non-Restaurant” store data listed first.

SELECT C.Category\_Name, P.PID

FROM CATEGORY AS C, SALE AS SA

INNER JOIN ASSIGNED AS A ON C.Category\_Name = A.Category\_Name

INNER JOIN PRODUCT AS P ON A.PID = P.PID

INNER JOIN SELL AS S ON P.PID = S.PID

INNER JOIN STORE AS ST ON S.Store\_Number = ST.Store\_Number

CASE ST.Has\_Restaurant

WHEN ST.Has\_Restaurant = TRUE THEN "Restaurant"

SUM(IFNULL(SA.Quantity, 0)) AS Quantity\_Sold

ELSE "Non-Restaurant" END AS Store\_Type

SUM(IFNULL(SA.Quantity, 0)) AS Quantity\_Sold

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 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 about quantity sold during and outside campaign for all products from DISCOUNT, PRODUCT, SALE, DAY, and ADVERTISING\_CAMPAIGN tables.
  + Get Discount\_Price data for all products (from the DISCOUNT table).
  + While a Product has Discount\_Price:
    - Get PID and Product\_Name data (from the PRODUCT table).

SELECT PRODUCT.PID, Product\_Name

FROM PRODUCT NATURAL JOIN DISCOUNT;

* + - Get Quantity data for this product sold during campaign (from SALE and ADVERTISING\_CAMPAIGN tables) at the specific discount Date (from DAY and DSICOUNT tables); Find total quantity sold during campaign by aggregating Quantity in all discount sale days that hold a campaign.
    - Get Quantity data for this product sold outside campaign (from SALE and ADVERTISING\_CAMPAIGN tables) at the specific discount Date (from DAY and DSICOUNT tables); Find total quantity sold outside campaign by aggregating Quantity in all discount sale days without a campaign.
    - Find 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 DIS.Discount\_Price, S.Quantity

FROM DISCOUNT AS DIS

INNER JOIN DAY AS D ON DIS.Date = D.Date

INNER JOIN PRODUCT AS P ON D.PID = P.PID

INNER JOIN SALE AS S ON D.Date = S.Date

INNER JOIN HOLD AS H ON D.Date = H.Date

INNER JOIN ADVERTISING\_CAMPAIGN AS A ON H.Description = A.Description

CASE H.Description

WHEN EXIST H.Description THEN "Sold During Campaign"

SUM(IFNULL(S.Quantity, 0)) AS Quantity\_Sold\_Camp

ELSE "Sold Outside Campaign" END

SUM(IFNULL(S.Quantity, 0)) AS Quantity\_Sold\_noCamp

(Quantity\_Sold\_Camp - Quantity\_Sold\_noCamp) AS Diff\_Quantity

GROUP BY P.PID

ORDER BY Diff\_Quantity DESC;

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