**Table of Contents**

[**Abstract Code**](#_wa5sa3868sir) **and SQL 1**

[**User Interface**](#_vwapcr69oq6i) **1**

[**View Category Report**](#_df30xbw9i42c) **2**

[**View Actual versus Predicted Revenue for Couches and Sofas**](#_ji4nykvoiklt) **3**

[**View Store Revenue by Year by State**](#_p6agvqllizzd) **5**

[**View Outdoor Furniture on Groundhog Day**](#_xr9lizrhlrxy) **6**

[**View State with Highest Volume for each Category**](#_vaxahjgb9ytf) **7**

[**View Revenue by Population**](#_3o0vtjv26awe) **8**

[**View Childcare Sales Volume**](#_u8savmc0ogbr) **9**

[**View Restaurant Impact on Category Sales**](#_mg1hj3a5p7nh) **9**

[**View Advertising Campaign Analysis**](#_np6f926aku6f) **10**

# 

# 

# Abstract Code

## User Interface

**Abstract Code:**

* Show ***“View Category Report”***,***“View Holiday”***, ***“Edit Holiday”***, ***“View City Population”***, ***“Edit City Population”***, ***“View Actual versus Predicted Revenue for Couches and Sofas Report”***, ***“View Store Revenue by Year by State”***, ***“View Outdoor Furniture on Groundhog Day”***, ***“View State with Highest Volume for each Category”***, ***“View Revenue by Population”***, ***“View Childcare Sales Volume”***, ***“View Restaurant Impact on Category Sales”***, ***“View Advertising Campaign Analysis”*** tabs.
* Upon:
  + Click ***View*** ***Category Report*** button- Jump to **View Category Report** task.
  + Click ***View Holiday*** button- Jump to **View Holiday** task.
  + Click ***Edit Holiday*** button- Jump to **Edit Holiday** task.
  + Click ***View City Population*** button- Jump to **View City Population** task.
  + Click ***Edit City Population*** button- Jump to **Edit City Population** task.
  + Click ***View Actual versus Predicted Revenue for Couches and Sofas*** button- Jump to **View Actual versus Predicted Revenue for Couches and Sofas** task.
  + Click ***View Store Revenue by Year by State*** button - Jump to **View Store Revenue by Year by State** task.
  + Click ***View Outdoor Furniture on Groundhog Day*** button- Jump to **View Outdoor Furniture on Groundhog Day** task.
  + Click ***View State with Highest Volume for each Category*** button- Jump to **View State with Highest Volume for each Category** task.
  + Click ***View State with Revenue by Population*** button- Jump to **View Revenue by Population** task.
  + Click ***View Childcare Sales Volume*** button- Jump to **View Childcare Sales Volume** task.
  + Click ***View Restaurant Impact on Category Sales*** button- Jump to **View Restaurant Impact on Category Sales** task.
  + Click ***View Advertising Campaign Analysis*** button- Jump to **View Advertising Campaign Analysis** task.

## View Category Report

**Abstract Code:**

* User clicked on ***View Category Report*** button from **Main Menu**:
* Run the **View Category Report** task: query for information about the category\_category name, total number of products in that category, and the average regular price (not including discount days) of all the products in that category.
* Sort the results by category name ascending.

|  |
| --- |
| SELECT Sales.quanlity\_sold AS quanlity\_sold, Product\_Category.category\_name AS CategoryName  FROM Sales  INNER JOIN Sales\_Product ON Sales.saleID = Sales\_Product.SaleID  INNER JOIN Product\_Category ON Product\_Category.PID = Sales\_Product.PID  GROUPBY CategoryName  ORDER BY CategoryName ASC; |

When ready, the user selects the next action from choices in the **Main Menu**.

## View Actual versus Predicted Revenue for Couches and Sofas

**Abstract Code:**

* User clicked on ***View Actual versus Predicted Revenue for Couches and Sofas*** button from **Main Menu**:
* Run **View Actual versus Predicted Revenue for Couches and Sofas** task: query for the information about each product in Couches and Sofas category.
  + Display PID, ProductName.

|  |
| --- |
| SELECT PID, product\_name  FROM Product  INNER JOIN Product\_Category ON Product\_Category.PID = Product.PID  WHERE Product\_Category.category\_name = “Couches and Sofas”; |

* + Find the current price using the PID; display DiscountPrice, RegularPrice.

|  |
| --- |
| SELECT Product.PID AS PID, DiscountPrice.discount\_price AS DiscountPrice, Product.regular\_price AS RegularPrice  FROM Product  INNER JOIN DiscountPrice ON Product.PID = DiscountPrice.PID  INNER JOIN Product\_Category ON Product\_Category.PID = Product.PID  WHERE Category.category\_name = “Couches and Sofas”; |

* + Find the total number of units ever sold
    - Calculate the total number of units sold at a discount (DiscountPrice\* QuantitySold)
    - Find the total number of units sold at regular price (RegularPrice\* QuantitySold)

|  |
| --- |
| SELECT PID, (TotalNumberOfUnitsSoldwithDiscountPrice + TotalNumberOfUnitsSoldwithRegularPrice) AS TotalNumberOfUnits  FROM(  (SELECT Sales\_Product.PID, SUM(Sales.quanlity\_sold) AS TotalNumberOfUnitsSoldwithDiscountPrice  FROM Sales  INNER JOIN Sales\_Product ON Sales.saleID = Sales\_Product.saleID  INNER JOIN DiscountPrice ON DiscountPrice.PID = Sales\_Product.PID  INNER JOIN Product\_Category ON Product\_Category.PID = Sales\_Product.PID  WHERE Category.category\_name = “Couches and Sofas”  GROUPBY Sales\_Product.PID) AS DiscountPriceSoldQuantity  JOIN (SELECT Sales\_Product.PID, SUM(Sales.quanlity\_sold) AS TotalNumberOfUnitsSoldwithRegularPrice  FROM Sales  INNER JOIN Sales\_Product ON Sales.saleID = Sales\_Product.saleID  INNER JOIN Product ON Product.PID = Sales\_Product.Product\_PID  INNER JOIN Product\_Category ON Product\_Category.PID = Product.PID  WHERE Category.category\_name = “Couches and Sofas”  GROUPBY Sales\_Product.PID  ) AS RegularPriceSoldQuantity  ON RegularPriceSoldQuantity.PID = DiscountPriceSoldQuantity.PID  ) temp  ; |

* + Find the actual revenue collected from all the sales of the product, the predicted revenue of the product on sale (based on 75% volume selling at regular price), and the difference between the actual revenue and the predicted revenue.
    - Calculate the actual revenue collected from all the sales of the product (RegularPrice/DiscountPrice \* QuantitySold)
    - Calculate the predicted revenue of the product on sale (based on 75% volume selling at retail price) (RegularPrice \* QuantitySold \* 75%)
    - Calculate the difference between the actual revenue and the predicted revenue

|  |
| --- |
| regular\_sold AS  (SELECT (Revenue1+Revenue2) AS Actual\_Revenue  FROM(  (SELECT SUM(Sales.quanlity\_sold\*DiscountPrice.discount\_price) AS Revenue1, YEAR(Date) AS Year  FROM Sales  INNER JOIN Sales\_Product ON Sales.saleID = Sales\_Product.saleID  INNER JOIN DiscountPrice ON DiscountPrice.PID = Sales\_Product.Product\_PID  INNER JOIN Product\_Category ON Product\_Category.PID = Sales\_Product.Product\_PID  WHERE Category.category\_name = “Couches and Sofas”) AS temp\_table\_A  JOIN  (SELECT SUM(Sales.quanlity\_sold\*Product.regular\_price) AS Revenue2,YEAR(Date) AS Year  FROM Sales  INNER JOIN Sales\_Product ON Sales.saleID = Sales\_Product.Sales\_SaleID  INNER JOIN Product ON Product.PID = Sales\_Product.Product\_PID  INNER JOIN Product\_Category ON Product\_Category.PID = Product.PID  WHERE Category.category\_name = “Couches and Sofas”) AS AS temp\_table\_B  ON AS temp\_table\_A.Year =AS temp\_table\_B.Year  ) temp  );  predicted\_sold AS  (SELECT SUM(Sales.quanlity\_sold\*Product.regular\_price\*75%) AS Predicted\_Revenue  FROM Sales  INNER JOIN Sales\_Product ON Sales.saleID = Sales\_Product.Sales\_SaleID  INNER JOIN Product ON Product.PID = Sales\_Product.Product\_PID  INNER JOIN Product\_Category ON Product\_Category.PID = Sales\_Product.Product\_PID  WHERE Category.category\_name = “Couches and Sofas”);  SELECT (Actual\_Revenue - Predicted\_Revenue) AS diff  FROM regular\_sold, predicted\_sold; |

When ready, the user selects the next action from choices in the **Main Menu**.

## 

## View Store Revenue by Year by State

**Abstract Code:**

* User clicked on ***View Store Revenue by Year by State*** button from **Main Menu**:
* Run the **View Store Revenue by Year by State** task: Display all the states in the Drop-down box.
  + Upon selection of the specified state
  + Display store\_ID, street\_address, city\_name

|  |
| --- |
| SELECT store\_ID, street\_address, city\_name  FROM Store; |

* + For each year:
    - Calculate the total revenue for all stores for each year in the selected state (revenue calculation must consider the products were sold at a discounted price).
    - Sort the total revenue by each year in ascending order and then by revenue in descending order.

|  |
| --- |
| SELECT (Revenue1+Revenue2) AS Revenue, Year  FROM  (  (SELECT SUM(Sales.quanlity\_sold\*DiscountPrice.Discount\_Price) AS Revenue1, YEAR(date) AS Year  FROM Sales  INNER JOIN Sales\_Product ON Sales.saleID = Sales\_Product.saleID  INNER JOIN DiscountPrice ON DiscountPrice.PID = Sales\_Product.PID  LEFT JOIN City ON City.city\_name = Sales.city\_name  WHERE City.state=$State AND YEAR(Sales.date) = $Year  GROUP BY YEAR(date)  ) AS DiscountRevenue  JOIN  (SELECT SUM(Sales.quanlity\_sold\*Product.regular\_price) AS Revenue2,YEAR(Date) AS Year  FROM Sales  INNER JOIN Sales\_Product ON Sales.saleID = Sales\_Product.Sales\_saleID  INNER JOIN Product ON Product.PID = Sales\_Product.PID  LEFT JOIN City ON City.city\_name = Sales.city\_name  WHERE City.state=$State AND YEAR(Sales.state) = $Year  GROUP BY YEAR(date)  ) AS RegularRevenue  ON DiscountRevenue.Year = RegularRevenue.Year  ) temp  ORDER BY YEAR ASC, Revenue DESC  ; |

When ready, the user selects the next action from choices in the **Main Menu**.

## View Outdoor Furniture on Groundhog Day

**Abstract Code:**

* User clicked on ***View Outdoor Furniture on Groundhog Day*** button from **Main Menu**:
* Run the **View Outdoor Furniture on Groundhog Day** task: query for information about the total number of outdoor furniture sold each year on Groundhog Day, average number of outdoor furniture sold per day each year, total number of outdoor furniture sold each year on Groundhog Day.
* Filter and display date (February 2) of each year.
  + Calculate the total number of units sold that year and daily average in the Outdoor Furniture category
  + Calculate the total number of units sold on Groundhog day that year in the Outdoor Furniture category
  + Sort the total number of units in the Outdoor Furniture category of each year in ascending order

|  |
| --- |
| WITH outdoor\_furniture\_sold AS  (SELECT quanlity\_sold  FROM Sales  INNER JOIN Sales\_Product ON Sales.saleID = Sales\_Product.saleID  INNER JOIN Product\_Category ON Product\_Category.PID = Sales\_Product.PID  WHERE Product\_Category.category\_name = “Outdoor Furniture”),  groudhogday\_furniture\_sold AS  (SELECT Year(Date) AS Year, SUM(quanlity\_sold) AS groundhog\_sold  FROM outdoor\_furniture\_sold  WHERE Month(Date) = 2 and Day(Date) = 2  GROUP BY Year),  total\_avg\_sold AS  (SELECT Year(Date) AS Year, SUM(quantity\_sold) AS total\_sold, (SUM(quanlity\_sold)/365) AS avg\_sold  FROM outdoor\_furniture\_sold)  GROUP BY Year),  SELECT groudhogday\_furniture\_sold.Year, groudhogday\_furniture\_sold.groundhog\_sold, total\_sold, avg\_sold  FROM total\_avg\_sold  LEFT JOIN groudhogday\_furniture\_sold ON groudhogday\_furniture\_sold.Year = total\_avg\_sold.Year  ORDER BY Year ASC; |

When ready, the user selects the next action from choices in the **Main Menu**.

## View State with Highest Volume for each Category

**Abstract Code:**

* User clicked on ***View State with Highest Volume for each Category*** button from **Main Menu**: query for information about the the states that sell greatest number of units in each category, all stores in that states at a selection month and year
* Filter and display the selected date

|  |
| --- |
| SELECT Year(Date) and Month(Date)  FROM Date |

* Run the **View State with Highest Volume for each Category task:**
  + Filter by each category
  + Calculate the total volume of units sold for that category of each state.
  + Select the state by finding the highest volume of units for that category of the state by states descending order
  + Select the stores in that state with the highest quantity sold in each category
  + User clicks ***drill-down detail*** button for category
  + Drop-down detail for each rows filtered by state, category, and date to discover which store in each city has the highest sales volume

|  |
| --- |
| WITH state\_sale as(  SELECT Product\_Category.category\_name as category\_name, SUM(Sales.quantity\_sold) as state\_total\_sold, City.state  FROM Store  INNER JOIN City ON City.city\_name = Store.city\_name  INNER JOIN Sales ON Store.store\_ID = Sales.store\_ID  INNER JOIN Sales-Product ON Sales.saleID = Sales\_Product.saleID  INNER JOIN Product\_Category ON Sales-Product.PID = Product\_Category.PID  WHERE Year(Date) = $Year and Month(Date) = $Month  GROUP BY City.state, Product\_Category.category\_name),  store\_sale as(  SELECT Store.store\_ID, Product\_Category.category\_name as category\_name, SUM(Sales.quantity\_sold) as store\_total\_sold  FROM Store  INNER JOIN Sales ON Store.store\_ID = Sales.store\_ID  INNER JOIN Sales\_Product ON Sales.saleID = Sales\_Product.saleID  INNER JOIN Product\_Category ON Sales\_Product.PID = Product\_Category.PID  WHERE Year(Date) = $Year and Month(Date) = $Month  GROUP BY Store.store\_ID, Product\_Category.category\_name);  SELECT category\_name, state  FROM state\_sale  LEFT JOIN store\_sale ON state\_sale.category\_name = store\_sale.category\_name  Where (category\_name, state\_total\_sold) IN (SELECT category\_name, MAX(state\_total\_sold) FROM state\_sale GROUP BY category\_name)  ORDER BY catogory\_name ASC; |

When ready, the user selects the next action from choices in the **Main Menu**.

## View Revenue by Population

**Abstract Code:**

* User clicked on***View Revenue by Population*** button from **Main Menu**: query for information about the revenue corresponds to cities with different sizes.
* Run the **View Revenue by Population** task:
  + Extract year from Date.date
  + For **each** CitySize level (Small, Medium, Large, Extra Large):
    - Calculate annual revenue for every year
    - Sort in ascending order：
      * years (oldest to newest)

CitySize (small, medium, large, extra large)

When ready, the user selects the next action from choices in the **Main Menu**.

|  |
| --- |
| SELECT  Year(Date)  CASE WHEN DiscountPrice.PID = PID AND Date IN (SELECT DiscountPrice.discount\_date FROM DiscountPrice) THEN DiscountPrice.discount\_price\*quantity\_sold ELSE regular\_price\*quantity\_sold END AS sale\_revenue  SUM(CASE WHEN city\_size = Small THEN sale\_revenue ELSE null END) AS small\_revenue,  SUM(CASE WHEN city\_size = Medium THEN sale\_revenue ELSE null END) AS med\_revenue,  SUM(CASE WHEN city\_size = Large THEN sale\_revenue ELSE null END) AS large\_revenue,  SUM(CASE WHEN city\_size = Extra Large THEN sale\_revenue ELSE null END) as xl\_revenue  FROM  ((((Store  INNER JOIN City ON Store.city\_name = City.city\_name) INNER JOIN Sales ON Store.storeID = Sale.storeID) INNER JOIN Date ON Sales.Date = Date.Date) INNER JOIN Product ON Sales\_Product.saleID = Sale.saleID)  GROUP BY Year  ORDER BY Year ASC; |

## View Childcare Sales Volume

**Abstract Code:**

* User clicked on***Childcare Sales Volume*** button from **Main Menu**:
* Run the **View Childcare Sales Volume**task:
  + Filter SalesDate to the last 12 months
  + Create ChildcareLimitCategory based on Store.ChildcareLimit as columns (When ChildcareLimit = 0, ChildcareLimitCategory = “No childcare”)
  + Extract Date.Month from Date.Date
  + for each Date.Month (as row) and ChildcareLimitCategory (as column):
    - Sum total sales

When ready, the user selects the next action from choices in the **Main Menu.**

|  |
| --- |
| SELECT \* FROM (  SELECT month(Date) AS month, childcare\_limit, (CASE WHEN DiscountPrice.PID = PID AND Date IN (SELECT DiscountPrice.discount\_date FROM DiscountPrice) THEN DiscountPrice.discount\_price\*quantity\_sold ELSE regular\_price\*quantity\_sold END) AS revenue  FROM (  (((Store INNER JOIN City ON Store.city\_name = City.city\_name) INNER JOIN Sales ON Store.storeID = Sale.storeID) INNER JOIN Date ON Sales.Date = Date.Date) INNER JOIN Product ON Sales\_Product.saleID = Sale.saleID)  WHERE year(Date.Date) = year(Now()) - 1  )  PIVOT (  SUM(revenue)  FOR childcare\_limit IN (SELECT childcare\_limit FROM City)  )  ORDER BY month ASC |

## View Restaurant Impact on Category Sales

**Abstract Code:**

* User clicked on***Restaurant Impact on Category Sales*** button from **Main Menu**:
* Run the **View Restaurant Impact on Category Sales**task: query for information about the product sold quantity in stores with or without onsite restaurant.
  + Calculate quantity sold in each store for each category
  + Sum total quantity sold in stores with restaurant for each category
  + Sum total quantity sold in stores with no restaurant for each category
  + Union two tables, order by CategoryName and Store\_Type

|  |
| --- |
| WITH store\_sale\_cat as(  SELECT Store.store\_ID, Product\_Category.category\_name as category\_name, SUM(Sales.quantity\_sold) as total\_sold  FROM Store  INNER JOIN Sales ON Store.store\_ID = Sales.store\_ID  INNER JOIN Sales\_Product ON Sales.saleID = Sales\_Product.saleID  INNER JOIN Product\_Category ON Sales\_Product.PID = Product\_Category.category\_name  GROUP BY Store.store\_ID, Product\_Category.category\_name)  SELECT \*  FROM (  SELECT category\_name, ‘Restaurant’ as store\_type, SUM(total\_sold) as quantity\_sold  FROM store\_sale\_cat  WHERE store\_ID IN (SELECT store\_ID FROM Store\_StoreAffiliates WHERE store\_affiliates = ‘‘Restaurant’ )  GROUP BY category\_name  UNION ALL  SELECT category\_name, ‘Non-Restaurant’ as store\_type, SUM(total\_sold) as quantity\_sold  FROM store\_sale\_cat  WHERE store\_ID NOT IN (SELECT store\_ID FROM Store\_StoreAffiliates WHERE store\_affiliates = ‘‘Restaurant’ )  GROUP BY category\_name  ) t  ORDER BY catogory\_name, store\_type; |

When ready, the user selects the next action from choices in the **Main Menu**.

## View Advertising Campaign Analysis

**Abstract Code:**

* User clicked on***Advertising Campaign Analysis*** button from **Main Menu**:
* Run the **View Advertising Campaign Analysis** task: query for information about the product sold quantity during ad or non ad time.
  + Filter only discount products, which are sold during Ad Campaign.
  + Calculate total item sold during campaign for discount product
  + Calculate total item sold during non-campaign period for discount product
  + Calculate sold-quantity difference for each discount product
  + Get top 10 and bottom 10 products of sold-quantity difference and return table order by difference in descending order

|  |
| --- |
| WITH sale\_item AS  (SELECT Product.PID, DiscountPrice.Date AS sale\_date, DiscountPrice.discount\_price AS sale\_price  FROM Product  INNER JOIN DiscountPrice ON DiscountPrice.PID = Product.PID  WHERE DiscountPrice.date in (SELECT date FROM AdCampaign WHERE ad\_campaign\_description IS NOT NULL)  ),  compaign\_sold AS  (SELECT sale\_items.PID, SUM(Sales.quantity\_sold) as total\_sale\_quantity  FROM Sales, Sales\_Product, sale\_items  WHERE Sales.saleID = Sales\_Product.saleID  AND Sales\_Product.ID = sale\_items.PID  AND Sales.date = sale\_items.sale\_date  GROUP BY sale\_items.PID),  regular\_sold AS  (SELECT sale\_items.PID, SUM(Sales.quantity\_sold) as total\_sale\_quantity  FROM Sales, Sales\_Product, sale\_items  WHERE Sales.saleID = Sales\_Product.saleID  AND Sales\_Product.PID = sale\_items.PID  AND Sales.date != sale\_items.sale\_date  GROUP BY sale\_items.PID),  Diff as  (SELECT compaign\_sold.PID, total\_sale\_quantity, total\_reg\_quantity (total\_sale\_quantity - total\_reg\_quantity) AS diff  FROM compaign\_sold, regular\_sold  WHERE compaign\_sold.PID = regular\_sold.PID)  SELECT \*  FROM(  SELECT Diff.PID, product\_name, total\_sale\_quantity, total\_reg\_quantity, difference  FROM Diff  JOIN Product ON Diff.PID = Product.PID  ORDER BY Diff.difference DESC LIMIT 10  UNION  SELECT Diff.PID, product\_name, total\_sale\_quantity, total\_reg\_quantity, difference  FROM Diff  JOIN Product ON Diff.PID = Product.PID  ORDER BY Diff.difference LIMIT 10  ) t  ORDER BY difference DESC; |

When ready, the user selects the next action from choices in the **Main Menu**.