Table of Contents:

Display Main Menu

Abstract Code

(Called upon successful login and upon ***Return to Main Menu*** button clicks.)

- When page is rendered, display output of:

```
SELECT COUNT(storeNumber) AS 'Number of Stores'
FROM Store;

SELECT COUNT(manufacturerName) AS 'Number of Manufacturers'
FROM Manufacturer;

SELECT COUNT(PID) AS 'Number of Products'
FROM Product;

SELECT COUNT(emailAddress) AS 'Number of Managers'
FROM Manager;

SELECT COUNT(DISTINCT emailAddress) AS 'Number of Active Managers'
FROM Manages;
```

- Show maintenance screen links (***Holiday Data***, ***Manager Profile***, ***Store Assignment***, and ***City Population***) on **Main Menu Screen*.***
- Show report screen links (***Manufacturers' Product Report***, ***Category Report***, ***Actual/Predicted GPS Report***, ***Store Revenue Report***, ***GH Day AC Report***, ***State Volume by Category Report***, and ***Revenue by Population Report***) on **Main Menu Screen**.
- Upon:
    - Click ***Holiday Data*** link: Jump to the Edit Holiday Data task.
    - Click ***Manager Profile*** link: Jump to the Edit Manager Profile task.
    - Click ***Store Assignment*** link: Jump to the Assign Stores task.
    - Click ***City Population*** link: Jump to the Update City Population task.
    - Click ***Manufacturers' Product Report*** link: Jump to the Display Manufacturer Summary task.
    - Click ***Category Report*** link: Jump to the Display Category Summary task.
    - Click ***Actual/Predicted GPS Report*** link: Jump to the Display Actual vs Predicted GPS task.
    - Click ***Store Revenue Report*** link: Jump to the Display Store Revenue by Year and State task.
    - Click ***GH Day AC Report*** link: Jump to the Display Air Conditioners on Groundhog Day task.
    - Click ***State Volume by Category Report*** link: Jump to the Display State Volume by Category Report task.
    - Click ***Revenue by Population Report*** link: Jump to the Display Annual Average Revenue by City Population task.

Edit Holiday Data

Abstract Code

(Called from the **Main Menu Screen** and below)

- Clear all input fields
- When page is rendered, display output of:

```
SELECT holidayDate, holidayName
FROM Holiday
ORDER BY holidayDate DESC;
```

- User enters **DATE** ($holidayDate, using CAST('$holidayDate' as DATE)) and **HOLIDAY NAME** ($holidayName) input fields.
  - Issue appropriate error message if entry is invalid.
- Upon:
  - Click ***Add Holiday*** button:
    - Insert new holiday information by running:

    ```
    INSERT INTO Holiday(holidayDate, holidayName)
    VALUES('$holidayDate', '$holidayDate');
    ```

    - If return shows success, display a success message. Otherwise, display an appropriate error message.
  - Click ***Return to Main Menu*** button: Call the Display Main Menu task.

Edit Manager Profile

Abstract Code

(Called from the **Main Menu Screen** and below)

- Display the **Manager Profiles Screen**.
- Clear all input fields and success/error messages
- When page is rendered, display output of:

```
SELECT managerName, emailAddress
FROM Manager
ORDER BY managerName;
```

- User enters **NAME** ($managerName) and/or **EMAIL ADDRESS** ($emailAddress) input fields.
- Upon:
  - ***Add Manager*** button click: Read the **NAME** ($managerName) and **EMAIL ADDRESS** ($emailAddress) input fields on the **Manager Profiles Screen**. If either field is empty or the email address is not valid, display an appropriate error message.

```
INSERT INTO Manager(emailAddress, managerName)
VALUES('$emailAddress','$managerName');
```

    - Insert a new manager by running:
    - If successful, display a success message, then call the Edit Manager Profile task. Otherwise, display an appropriate error message.
  - ***Update Manager*** button click: Read the **NAME** ($managerName) and **EMAIL ADDRESS** ($emailAddress) input fields on the **Manager Profiles Screen**. If either field is empty or the email address is not valid, display an appropriate error message. Update the manager information via:

```
UPDATE Manager
SET managerName = '$managerName'
WHERE emailAddress = '$emailAddress';
```

  - ***Deactivate Manager*** button click:
    - Clear the output area and any success/error messages on the **Manager Profiles Screen**.
    - Read the **EMAIL ADDRESS** ($emailAddress) input fields on the **Manager Profiles Screen**. If the email address is not valid, display an appropriate error message.
    - Delete all store assignments for the manager by running:

```
DELETE FROM Manages
WHERE emailAddress = '$emailAddress';
```

    - If the return shows success, display a success message; otherwise display an appropriate error message.

- ***Delete Manager*** button click:
  - Clear the output area and any success/error messages on the **Manager Profiles Screen**.
  - Read the **EMAIL ADDRESS** (`$emailAddress`) input field on the **Manager Profiles Screen**. If the email address is not valid, display an appropriate error message.
  - Check to see if the manager has any store assignments by running:

    ```
    SELECT storeNumber, emailAddress
    FROM Manages
    WHERE emailAddress= '$emailAddress'
    ```

  - If this returns any results, display an appropriate error message saying that the manager cannot be deleted because the manager is currently assigned to stores. Otherwise, delete the manager by running:

    ```
    DELETE FROM Manager
    WHERE emailAddress = '$emailAddress';
    ```

  - If return shows success, display a success message. Otherwise, display an appropriate error.
    - Otherwise, write an appropriate error message.
- ***Return to Main Menu*** button click: Call the Display Main Menu task.

Assign Stores

Abstract Code

(Called from the **Main Menu Screen**)

- Display the **Store Assignment Screen**.
- User enters **STORE NUMBER** ($storeNumber) and/or **EMAIL ADDRESS** ($emailAddress) input fields.
- Upon:
  - ***View Manager Assignments*** button click:
    - Clear the output area and any success/error messages on the **Store Assignment Screen**.
    - Read the **EMAIL ADDRESS** ($emailAddress) input field on the **Store Assignment Screen**. If the email address is not valid, display an appropriate error message.
    - Display the manager's current store assignments by displaying the output of:

```
SELECT storeNumber
FROM Manages
WHERE emailAddress = '$emailAddress'
ORDER BY storeNumber
```

    - If no results are returned, display a message that the manager has no store assignments. Otherwise display a success message.
  - ***View Store Assignments*** button click:
    - Clear the output area and any error messages on the **Store Assignment Screen**.
    - Read the **STORE NUMBER** ($storeNumber) input field on the **Store Assignment Screen**.
      - If the entry is not a valid integer, display an error message and prompt for re-entry until a valid integer is entered.
    - Display the store's current manager assignments by displaying the output of:

```
SELECT emailAddress
FROM Manages
WHERE storeNumber = '$storeNumber'
ORDER BY emailAddress
```

    - If no results are returned, display a message that the store has no manager assignments. Otherwise display a success message.
  - ***Assign Manager to Store*** button click:
    - Clear the output area and any success/error messages on the **Store Assignment Screen**.
    - Read the **STORE NUMBER** ($storeNumber) and **EMAIL ADDRESS** ($emailAddress) input fields on the **Store Assignment Screen**.
      - If $storeNumber is not a valid integer or $emailAddress Is not a valid email address, display an appropriate error message and prompt to re-enter until valid entries are made.
    - Assign the manager to the store by running:

```
INSERT INTO Manages(storeNumber, emailAddress)
VALUES('$storeNumber', '$emailAddress');
```

    - If return indicates success, display a success message. Otherwise, display an appropriate error message.

Table of Contents

- ***Unassign Manager from Store*** button click:
  - Clear the output area and any success/error messages on the **Store Assignment Screen**.
  - Read the **STORE NUMBER** ($storeNumber) and **EMAIL ADDRESS** ($emailAddress) input fields on the **Store Assignment Screen**.
    - If $storeNumber is not a valid integer or $emailAddress Is not a valid email address, display an appropriate error message and prompt to re-enter until valid entries are made.
  - Remove the manager assignment from the store by running:

```
DELETE FROM Manages
WHERE
    emailAddress = '$emailAddress' AND
    storeNumber = '$storeNumber';
```

  - If return indicates success, display a success message. Otherwise, display an appropriate error message.
- ***Return to Main Menu*** button click: Call the Display Main Menu task.

Update City Population

Abstract Code

(Called from the **Main Menu Screen**)

- Display **City Population Screen**
- When page is rendered, display output of:

```
SELECT cityName, state, population
FROM City
ORDER BY state, cityName;
```

- User may make entries in **CITY NAME** ($cityName), **STATE** ($state), and **POPULATION** ($population) input fields.
- Upon:
    - ***Update*** button clicked:
        - Clear all success and error messages.
        - Check that the **CITY NAME** and **STATE** input field entries are non-empty and that the **POPULATION** input field holds a valid integer value: If not, display an appropriate error message and prompt to re-enter until valid values are entered.
        - Update the population by running:

```
UPDATE City
    SET Population = '$population'
    WHERE cityName = '$cityName' AND State = '$state';
```

        - If return indicates success, display a success message. Otherwise, display an appropriate error message.
    - ***Return to Main Menu*** button click: Call the Display Main Menu task.

Display Manufacturer Summary

Abstract Code

(Called from the **Main Menu Screen**)

- Display **Manufacturers' Product Report Screen**.
- When page is rendered, combine the output of the following SQL with hyperlinks for each line, associating each line's text and manufacturer name (**Manufacturer**.*Name*) with the hyperlink:

```
SELECT
    manufacturerName,
    COUNT(PID),
    ROUND(AVG(price), 2),
    MIN(price),
    MAX(price)
FROM Product
GROUP BY manufacturerName
ORDER BY AVG(price) DESC
LIMIT 100;
```

- Upon:
    - ***Drill-Down Link*** click: Run the Display Manufacturer Drill-Down Details task, sending it the text and **Manufacturer**.*Name* associated with the clicked hyperlink.
    - ***Return to Main Menu*** button click: Run the Display Main Menu task.

Display Manufacturer Drill-Down Details

Abstract Code

(Called from the **Manufacturers' Product Report Screen** by the Display Manufacturer Summary task)

- Display the **Manufacturer Drill-Down Details Screen**.
- Retrieve the **Manufacturer**.*Name* ($manufacturerName) and text ($text) passed from the Display Manufacturer Summary task.
- Use the output of the following SQL to obtain the manufacturer's maximum discount ($maximumDiscount), and use it along with $manufacturerName and $text to display that information in the header area of the screen.

```
SELECT maximumDiscount
FROM Manufacturer
WHERE manufacturerName = '$manufacturerName ';
```

- Display in the details area of the **Manufacturer Drill-Down Details Screen** the output of:

```
SELECT
    Product.PID,
    productName,
    GROUP_CONCAT(DISTINCT categoryName SEPARATOR ', '),
    price
FROM
    Product,
    CategorizedBy
WHERE
    CategorizedBy.PID = Product.PID AND
    manufacturerName = '$manufacturerName'
GROUP BY Product.PID
ORDER BY price DESC;
```

- Upon:
    - ***Return to Manufacturers' Product Report*** button click: close the **Manufacturer Drill-Down Details Screen** and run Display Manufacturer Summary task.
    - ***Return to Main Menu*** button click: run Display Main Menu task.

Display Category Summary

Abstract Code

(Called from the **Main Menu Screen** and below)

- Display **Category Report Screen**
- Display the output of the following SQL:

```
SELECT
    C.categoryName AS Category,
    COUNT(CB.PID) AS `Number of Products`,
    COUNT(DISTINCT P.manufacturerName) AS `Number of Manufacturers`,
    ROUND(AVG(P.price), 2) AS `Average Price`
FROM Category C
JOIN
    CategorizedBy CB
ON C.categoryName = CB.categoryName
JOIN
    Product P
ON CB.PID = P.PID
GROUP BY C.categoryName
ORDER BY C.categoryName;
```

- Upon ***Return to Main Menu*** button click, call the Display Main Menu task.

Display Actual vs Predicted GPS

Abstract Code

(Called from the **Main Menu Screen**)

- Display the **GPS Report Screen**.
- Display the output from the following SQL:

```
SELECT
    `Product ID`,
    `Product Name`,
    `Retail Price`,
    SUM(`Total Quantity`) AS 'Total Units Sold',
    SUM(`Sale Quantity`) AS 'Total Units Sold at Discount',
    SUM(`Retail Quantity`) AS 'Total Units Sold at Retail',
    SUM(`Transaction Amount`) AS 'Actual Revenue',
    ROUND(SUM(`Predicted Amount`), 2) AS 'Predicted Revenue',
    ROUND(SUM(Difference), 2) AS 'Difference'
FROM
    (SELECT
        S.PID AS 'Product ID',
        P.productName AS 'Product Name',
        P.price AS 'Retail Price',
        IFNULL(G.salePrice, 0) AS 'Sale Price',
        C.categoryName,
        storeNumber,
        transactionDate,
        SUM(IF(ISNULL(G.salePrice), 0, quantity)) AS 'Sale Quantity',
        SUM(IF(ISNULL(G.salePrice), quantity, 0)) AS 'Retail Quantity',
        SUM(quantity * IF(ISNULL(G.salePrice), P.price, G.salePrice))
            AS 'Transaction Amount',
        SUM(IF(ISNULL(G.salePrice), quantity * P.price, 0.75 * quantity *
            P.price)) AS 'Predicted Amount',
        quantity * IF(ISNULL(G.salePrice), P.price, G.salePrice) -
            SUM(IF(ISNULL(G.salePrice), quantity * P.price, 0.75 * quantity *
            P.price)) AS 'Difference',
        SUM(quantity) AS `Total Quantity`
    FROM StoreSellsProduct S
    JOIN CategorizedBy C
        ON S.PID = C.PID AND C.categoryName = "GPS"
    JOIN Product P
        ON S.PID = P.PID
    LEFT OUTER JOIN GoesOnSale G
        ON S.PID = G.PID AND transactionDate = saleDate
    GROUP BY S.PID, P.productName, P.price, salePrice, C.categoryName,
        storeNumber, transactionDate
    ) AS InnerSelect
GROUP BY `Product ID`, `Product Name`, `Retail Price`
HAVING ABS(Difference) > 5000
ORDER BY `Difference` DESC, `Product ID`;
```

- Upon ***Return to Main Menu*** button click, call the Display Main Menu task.

Display Store Revenue by Year and State

Abstract Code

(Called from the **Main Menu Screen**)

- Display the **Store Revenue by Year and State Report Screen**
- Fill state dropdown box with distinct states, ordered ascending, with the results of the following:

```
SELECT DISTINCT state
FROM City
ORDER BY state;
```

- User may select state from dropdown box ($state).
- Upon:
  - ***Generate Report*** button clicked:
    - Display the results from the following SQL:

```
SELECT
    Store.storeNumber,
    Store.streetAddress,
    City.cityName,
    YEAR(StoreSellsProduct.transactionDate) AS Year,
    SUM(IF(StoreSellsProduct.transactionDate = GoesOnSale.saleDATE AND
        StoreSellsProduct.PID = GoesOnSale.PID,GoesOnSale.salePrice, Product.price)
        * StoreSellsProduct.quantity) AS Revenue
FROM
    StoreSellsProduct
LEFT OUTER JOIN
    GoesOnSale
ON
    StoreSellsProduct.transactionDate = GoesOnSale.saleDATE AND
        StoreSellsProduct.PID = GoesOnSale.PID,
    Product,
    Store,
    City
WHERE City.state = '$state'
AND Store.state = '$state'
AND Store.cityName = City.cityName
AND StoreSellsProduct.storeNumber = Store.storeNumber
AND StoreSellsProduct.PID = Product.PID
GROUP BY
    Store.storeNumber,
    Store.streetAddress,
    City.cityName,
    YEAR(StoreSellsProduct.transactionDate)
ORDER BY Year, Revenue DESC;
```

  - ***Return to Main Menu*** button click, call the Display Main Menu task.

Table of Contents

-

Display Air Conditioners on Groundhog Day

Abstract Code

(Called from the **Main Menu Screen**)

- Display the **Air Conditioners on Groundhog Day Report Screen**.
- Upon:
  - ***Generate Report*** button clicked: Display the results of the following SQL:

```
SELECT
    YEAR(StoreSellsProduct.transactionDate) AS Year,
    SUM(StoreSellsProduct.quantity) AS `Total AC Items Sold`,
    ROUND(SUM(StoreSellsProduct.quantity)/365, 0) AS
        `Average (Rounded) Number of Units Sold/Day`,
    GH.Total AS `Total Units Sold on GroundHog Day`
FROM
    StoreSellsProduct,
    CategorizedBy,
    (SELECT
        SUM(StoreSellsProduct.quantity) AS Total,
        YEAR(StoreSellsProduct.transactionDate) AS Year
    FROM
        StoreSellsProduct,
        CategorizedBy
    WHERE
        Categorizedby.categoryName = 'air conditioning' AND
        StoreSellsProduct.PID = Categorizedby.PID AND
        DATE_FORMAT(StoreSellsProduct.transactionDate,'%m-%d') =
            '02-02'
    GROUP BY YEAR(StoreSellsProduct.transactionDate)
    ) AS GH
WHERE
    Categorizedby.categoryName = 'air conditioning' AND
    StoreSellsProduct.PID = Categorizedby.PID AND
    GH.Year = YEAR(StoreSellsProduct.transactionDate)
GROUP BY
    YEAR(StoreSellsProduct.transactionDate),
    GH.Total
ORDER BY YEAR ASC;
```

  - ***Return to Main Menu*** button click, call the Display Main Menu task.

Display State Volume by Category

Abstract Code

(Called from the **Main Menu Screen** and below)

- Display **State Volume by Category Report Screen** and Populate the month/year dropdown from:

```
SELECT
    DISTINCT DATE_FORMAT(StoreSellsProduct.transactionDate,'%m-%Y')
FROM StoreSellsProduct
ORDER BY
    DATE_FORMAT(StoreSellsProduct.transactionDate,'%m-%Y') DESC;
```

- User will select a month/year from the dropdown list.
- Upon:
  - ***Generate Report*** button click:
    - Clear any previously written lines.
    - Read the selected month ($month) and year ($year) from the dropdown list.
    - With a hyperlink associated with each category and state, display results of the following SQL:

```
SELECT Category, State, Units
FROM
    (SELECT unitsCategory AS Category, MAX(units) AS Units
    FROM
        (SELECT CAT.categoryName AS unitsCategory, City.state, SUM(quantity) AS units
        FROM StoreSellsProduct SSP
        JOIN Product P ON P.PID = SSP.PID
        JOIN Store S ON SSP.storeNumber = S.storeNumber
        JOIN City ON City.cityName = S.cityName AND City.state = S.state
        JOIN CategorizedBy CB ON CB.PID = SSP.PID
        JOIN Category CAT ON CAT.categoryName = CB.categoryName
        WHERE
            MONTH(SSP.transactionDate) = '$month' AND
            YEAR(SSP.transactionDate) = '$year'
        GROUP BY CAT.categoryName, City.state) AS Q1
    GROUP BY unitsCategory) AS Q2
    JOIN
    (SELECT
        CAT.categoryName AS stateCategory,
        City.state AS State,
        SUM(quantity) AS stateUnits
        FROM StoreSellsProduct SSP
    JOIN Product P ON P.PID = SSP.PID
    JOIN Store S ON SSP.storeNumber = S.storeNumber
    JOIN City ON City.cityName = S.cityName AND City.state = S.state
    JOIN CategorizedBy CB ON CB.PID = SSP.PID
    JOIN Category CAT ON CAT.categoryName = CB.categoryName
    WHERE
        MONTH(SSP.transactionDate) = '$month' AND
        YEAR(SSP.transactionDate) = '$year'
    GROUP BY CAT.categoryName, City.state) AS Q3
    ON Category = stateCategory AND Units = stateUnits
ORDER BY Category;
```

Table of Contents

- Upon click of any of the hyperlinks:
  - Clear the drill-down area provided at the bottom of the **State Volume by Category Report Screen**.
  - Read the selected month ($month) and year ($year) from the dropdown list.
  - Determine the category ($category) and state ($state) associated with the clicked hyperlink.
  - Display the category ($category), year($year), month ($month), and state ($state) in the header area.
  - Display the results of the following SQL in the drill-down area:

```
SELECT
    Store.storeNumber AS `Store Number`,
    streetAddress AS `Address`,
    cityName AS City,
    IFNULL(managerName, '[No manager assigned]') AS Manager,
    Manager.emailAddress AS Email
FROM
    Store
LEFT JOIN
    Manages
ON Manages.storeNumber = Store.storeNumber
LEFT JOIN
    Manager
ON Manager.emailAddress = Manages.emailAddress
WHERE Store.state = '$state'
ORDER BY Store.storeNumber, managerName;
```

- ***Return to Main Menu*** button click, call the Display Main Menu task.

Display Annual Average Revenue by City Population

Abstract Code

(Called from the **Main Menu Screen**)

- Display the **Annual Average Revenue by City Population Report Screen**.
- When page is rendered, display output of:

```
SELECT
    YEARS.`Year`,
    ROUND(IFNULL(smallCategory/smallCount, 0), 2) AS `Small Category`,
    ROUND(IFNULL(mediumCategory/mediumCount, 0), 2) AS `Medium Category`,
    ROUND(IFNULL(largeCategory/largeCount, 0), 2) AS `Large Category`,
    ROUND(IFNULL(extraLargeCategory/extraLargeCount, 0), 2) AS `Extra Large Category`
FROM
    (SELECT DISTINCT YEAR(transactionDate) AS `Year`
    FROM StoreSellsProduct SSP) AS YEARS
    LEFT JOIN
        (SELECT
            YEAR(transactionDate) AS `Year`,
            ROUND(SUM(quantity * IF(ISNULL(GOS.salePrice), P.price, GOS.salePrice)),
                2) AS `smallCategory`,
            COUNT(C.cityName) AS `smallCount`
        FROM StoreSellsProduct SSP
        JOIN Product P
        ON P.PID = SSP.PID
        JOIN Store S
        ON S.storeNumber = SSP.storeNumber
        JOIN City C
        ON C.cityName = S.cityName AND C.state = S.state AND C.population < 3700000
        LEFT OUTER JOIN GoesOnSale GOS
        ON SSP.transactionDate = GOS.saleDate AND SSP.PID = GOS.PID
        GROUP BY YEAR(transactionDate)) AS SMALL
    ON YEARS.`Year` = SMALL.`Year`
LEFT JOIN



.
.
.




[Continued on next page]
```

[Continued on next page]

[continued from previous page]

```
(SELECT
    YEAR(transactionDate) AS `Year`,
    ROUND(SUM(quantity * IF(ISNULL(GOS.salePrice), P.price, GOS.salePrice)),
        2) AS `mediumCategory`,
    COUNT(C.cityName) AS `mediumCount`
    FROM StoreSellsProduct SSP
    JOIN Product P
    ON P.PID = SSP.PID
    JOIN Store S
    ON S.storeNumber = SSP.storeNumber
    JOIN City C
    ON C.cityName = S.cityName AND C.state = S.state AND
        C.population < 6700000 AND C.population >= 3700000
    LEFT OUTER JOIN GoesOnSale GOS
    ON SSP.transactionDate = GOS.saleDate AND SSP.PID = GOS.PID
GROUP BY YEAR(transactionDate)) AS MEDIUM
ON YEARS.`Year` = MEDIUM.`Year`
LEFT JOIN
(SELECT
    YEAR(transactionDate) AS `Year`,
    ROUND(SUM(quantity * IF(ISNULL(GOS.salePrice), P.price, GOS.salePrice)),
        2) AS `largeCategory`,
    COUNT(C.cityName) AS `largeCount`
    FROM StoreSellsProduct SSP
    JOIN Product P
    ON P.PID = SSP.PID
    JOIN Store S
    ON S.storeNumber = SSP.storeNumber
    JOIN City C
    ON C.cityName = S.cityName AND C.state = S.state AND
        C.population < 9000000 AND C.population >= 6700000
    LEFT OUTER JOIN GoesOnSale GOS
    ON SSP.transactionDate = GOS.saleDate AND SSP.PID = GOS.PID
GROUP BY YEAR(transactionDate)) AS LARGE
ON YEARS.`Year` = LARGE.`Year`
LEFT JOIN
(SELECT
    YEAR(transactionDate) AS `Year`,
    ROUND(SUM(quantity * IF(ISNULL(GOS.salePrice), P.price, GOS.salePrice)),
        2) AS `extraLargeCategory`,
    COUNT(C.cityName) AS `extraLargeCount`
    FROM StoreSellsProduct SSP
    JOIN Product P
    ON P.PID = SSP.PID
    JOIN Store S
    ON S.storeNumber = SSP.storeNumber
    JOIN City C
    ON C.cityName = S.cityName AND C.state = S.state AND
        C.population >= 9000000
    LEFT OUTER JOIN GoesOnSale GOS
    ON SSP.transactionDate = GOS.saleDate AND SSP.PID = GOS.PID
GROUP BY YEAR(transactionDate)) AS EXTRALARGE
ON YEARS.`Year` = EXTRALARGE.`Year`
ORDER BY YEARS.`Year`;
```

Table of Contents

- Upon:
    - ***Return to Main Menu*** button click: run Display Main Menu task.