Table of Contents

Login

Abstract Code

- User enters employee id and password (last 4 ssn + "-" + LastName) input fields
- User's inputs are validated to ensure they were entered in the right format, and this validation is run in client side
- Once validation is successful for both input fields' values, user's inputs are stored in $employeeID and $password variables in client side
- If data validation is successful, then:
    - When ***Login*** button is clicked:

      ** Developer's note: This SQL runs in server side to get a user's employee ID from User table for a given $employeeID and $password.

      ```
      SELECT EmployeeID FROM User
      WHERE EmployeeID = '$employeeID' AND '$password'
      =
      ( SELECT LastFourSSN + '-' + LastName FROM User WHERE EmployeeID = '$employeeID');
      ```

    - If User record is found, then:
        - Store $employeeID value in a session variable called $session.employeeID.
        - Go to **Main Menu** form.
    - Else:
        - Display an error message "User is not found."
        - Go to **Main Menu** form.
- Else employee id and/or password input fields are invalid
    - Stay in **Login** form, display an error message and highlight the field whose value was invalid

Main Menu

Abstract Code

- Show "***Manufacturer's Product Report***", "***Category Report***", "***Actual versus Predicted Revenue For GPS units***", ***"Air Conditioners on Groundhog Day?"***, ***"Store Revenue by Year by State***", "***District with Highest Volume for each Category"***, "***Revenue by Population***", ***"Add Holiday"***, ***"View Audit Log"***, ***"View Holiday"*** , and "***Log Out***" tabs in the navigation bar.
- Upon:
    - Click ***Manufacturer's Product Report*** tab – Jump to **View Manufacturer's Product Report** task
    - Click ***Category Report*** tab – Jump to **View Category Reports** task
    - Click ***Actual versus Predicted Revenue For GPS units*** tab – Jump to **View *Actual versus Predicted Revenue For GPS units* Reports** task
    - Click ***Air Conditioners on Groundhog Day?*** tab – Jump to **View *Air Conditioners on Groundhog Day?* Reports** task
    - Click ***Stores Revenue by Year by State*** tab – Jump to **View *Stores Revenue by Year by State* Reports** task
    - Click ***District with Higher Volume for each Category*** tab – Jump to **View *District with Higher Volume for each Category* Reports** task
    - Click ***Revenue by Population*** tab – Jump to **View *Revenue by Population* Reports** task
    - Click ***Add Holiday*** tab – Jump to **Add Holiday** task
    - Click ***View Audit Log*** tab – Jump to **View Audit Log** task
    - Click ***View Holidays*** tab – Jump to **View Holiday** task
    - Click ***Log Out*** tab – Invalidate login session variable and go back to the **Login** form.

- Query for information about the Count of Stores, Holidays, Cities, Districts, Manufacturers, Products, Categories, Holidays
- Find the Current User using the User.EmployeeID and a session variable; Display users First and LastName

  ** Developer's note: $session.employeeID is here given with a value stored in a client's session variable.

```
SELECT FirstName, LastName FROM User WHERE User.EmployeeID =
'$session.employeeID';
```

- Find total number of Stores; Display total number of Stores

```
SELECT Count(StoreNumber) AS Stores_Total FROM Store;
```

- Find total number of stores in each city; Display total number of stores

```
SELECT CityName, Count(StoreNumber) AS Stores_By_City FROM Store GROUP BY
CityName;
```

- Find Holiday Name and Date using Date; Display Holiday Name and Date

```
SELECT HolidayName, BusinessDate from Holidays ORDER BY BusinessDate;
```

- Find Store City, District and Phone Number using StoreNumber and cityName; Display StoreNumber, CityName, PhoneNumber, District of Each Store

```
SELECT Store.StoreNumber, Store.PhoneNumber, Store.DistrictNumber,
City.CityName, City.State
FROM Store, City
WHERE Store.CityName = City.CityName
ORDER BY Store.StoreNumber;
```

- Find ProductID, retailPrice, ProductName, ManufacturerName, and Category using ManufacturerName and ProductID; Display Manufacturer, Category, Product Name, ProductID, and RetailPrice for each product

```
SELECT Product.PID, Product.RetailPrice, Product.ProductName,
Manufacturer.ManufacturerName, Assignto.CategoryName as CategoryName
FROM Product, Manufacturer, Assignto
WHERE Product.ManufacturerName = Manufacturer.ManufacturerName AND
Product.PID = Assignto.PID
ORDER BY Product.PID ;
```

View Holidays

Abstract Code

- Check if user has been granted to access to all districts
- If all districts are accessible, then:
    - View all holidays as a list in a data grid.

```
SELECT HolidayName, BusinessDate FROM Holidays ORDER BY BusinessDate;
```

    - Display ***Add Holiday*** button on top of the form
- Else:
    - Display an error message
    - Go back to **Main Menu** form

Add Holidays

Abstract Code

- Check if user has been granted to access to all districts
- If all districts are accessible, then:
    - Enter the edit mode in UI to let a user add a new holidayDisplay ***Save Holiday*** button next
    - If ***Save Holiday*** button is clicked then:

```
INSERT INTO Holidays
(HolidayName, Date)
VALUES ('$HolidayName', '$Date');
```

- Else:
    - Display an error message
    - Go back to **Main Menu** form

View Manufacturers Product report

Abstract Code:

- Run the View Manufacturers Product Report task
    - Query manufacturers by manufacturerName and display manufacturer name, count of products offered, average retail price of all manufacturer products, minimum retail price, maximum retail price
    - Sort by manufacturerName
    - Order by AveragePrice, Descending

```
SELECT Manufacturer.ManufacturerName, COUNT(Product.PID) AS
ProductCount, AVG(Product.RetailPrice) AS AveragePrice,
MAX(Product.RetailPrice), MIN(Product.RetailPrice)
FROM Manufacturer JOIN Product ON Product.ManufacturerName =
Manufacturer.ManufacturerName
GROUP BY Manufacturer.ManufacturerName
ORDER BY AveragePrice DESC;
```

- Click on Manufacturer to view details
    - Query manufacturer by manufacturerName and display product, ProductID, productName, Product Category(ies), productPrice
    - Order by product Price descending

```
SELECT Product.ProductName, Product.PID, Product.RetailPrice,
Assignto.CategoryName
FROM Product, Manufacturer, Assignto
WHERE Product.ManufacturerName = Manufacturer.ManufacturerName AND
Product.PID = Assignto.PID
ORDER BY Product.RetailPrice DESC;
```

- Generate AuditLogEntry

  Developer's Note: '$EmployeeID' is acquired from the session. '$TimeStamp' is dynamically set to the current time, and '$ReportName' reflects the name of the report currently being viewed by the user.

  - Insert AuditLog to AuditLogEntry

```
INSERT INTO AuditLogEntry (EmployeeID, Timestamp, ReportName)
VALUES ('$EmployeeID', '$TimeStamp', 'ReportName');
```

View Category Report

Abstract Code:

- Run View Category Report task
  - Query Categories by CategoryName and display CategoryName, count of products, count of manufacturers, average retail price
  - Sort by categoryName Ascending

```
SELECT Assignto.CategoryName, COUNT(DISTINCT Product.PID),
COUNT(DISTINCT Product.ManufacturerName), AVG(Product.RetailPrice)
FROM Assignto JOIN Product ON Assignto.PID = Product.PID
GROUP BY Assignto.CategoryName
ORDER BY Assignto.CategoryName ASC;
```

- Generate AuditLogEntry

  Developer's Note: '$EmployeeID' is acquired from the session. '$TimeStamp' is dynamically set to the current time, and '$ReportName' reflects the name of the report currently being viewed by the user.

  - Insert AuditLog to AuditLogEntry

```
INSERT INTO AuditLogEntry (EmployeeID, Timestamp, ReportName)
VALUES ('$EmployeeID', '$TimeStamp', 'ReportName');
```

View Actual versus Predicted Revenue for GPS units Report

Abstract Code:

- Find the Current User using the User.EmployeeID and Query Districts Associated with User
- Query Product, Sells, Assignto to get TotalSales of each Product in GPS category
- Query Product, Sells, Discounted to get a count of all Discounted Sales by Product

- Query TotalSales, DiscountedSales to calculate Actual revenue received from units sold without discounts

- Query TotalSales, DiscountedSales, NonDiscountedSales to Calculate PredictedRevenue and Difference Between Predicted and Actual Revenue
- Query Predicted Revenue and Display difference between actual and predicted revenue where predicted revenue differences > 200 (positive or negative), ordered by Revenue Difference descending

```
SELECT
  TS.PID,
  TS.ProductName,
  TS.RetailPrice,
  TS.TotalUnitsSold,
  CASE WHEN DS.DiscountUnitsSold IS NULL THEN 0 ELSE DS.DiscountUnitsSold END
AS DiscountedUnitsSold,
  TS.TotalUnitsSold - CASE WHEN DS.DiscountUnitsSold IS NULL THEN 0 ELSE
DS.DiscountUnitsSold END AS NonDiscountedUnitsSold,
  TS.TotalUnitsSold * TS.RetailPrice - CASE WHEN DS.DiscountedRevenue IS NULL
THEN 0 ELSE DS.DiscountedRevenue END AS ActualRevenue,
  TS.TotalUnitsSold * TS.RetailPrice * 0.75 AS PredictedRevenue,
  (TS.TotalUnitsSold * TS.RetailPrice - CASE WHEN DS.DiscountedRevenue IS NULL
THEN 0 ELSE DS.DiscountedRevenue END) - (TS.TotalUnitsSold * TS.RetailPrice * 0.75)
AS RevenueDifference
FROM (
  SELECT
    Product.PID,
    Product.ProductName,
    Product.RetailPrice,
    SUM(Sells.QuantitySold) AS TotalUnitsSold
  FROM
    Product
    JOIN Sells ON Product.PID = Sells.PID
    JOIN Assignto ON Product.PID = Assignto.PID
  WHERE
    Assignto.CategoryName = 'GPS'
  GROUP BY
    Product.PID, Product.ProductName, Product.RetailPrice
) TS
LEFT JOIN (
  SELECT
    Discount.PID,
    SUM(Sells.QuantitySold) AS DiscountUnitsSold,
    SUM(Sells.QuantitySold * Discount.DiscountPrice) AS DiscountedRevenue
```

```
    FROM
        Discount
        JOIN Sells ON Discount.PID = Sells.PID AND Discount.BusinessDate =
Sells.DateSold
    GROUP BY
        Discount.PID
) DS ON TS.PID = DS.PID
WHERE
    ABS((TS.TotalUnitsSold * TS.RetailPrice - CASE WHEN DS.DiscountedRevenue IS
NULL THEN 0 ELSE DS.DiscountedRevenue END) - (TS.TotalUnitsSold * TS.RetailPrice *
0.75)) > 200
ORDER BY
    RevenueDifference DESC;
```

- Generate AuditLogEntry

  Developer's Note: '$EmployeeID' is acquired from the session. '$TimeStamp' is dynamically set to the current time, and '$ReportName' reflects the name of the report currently being viewed by the user.

  - Insert AuditLog to AuditLogEntry

```
INSERT INTO AuditLogEntry (EmployeeID, Timestamp, ReportName)
VALUES ('$EmployeeID', '$TimeStamp', 'ReportName');
```

View Air Conditioners on Groundhog Day? Report

Abstract Code:

- Generate Report Table

- Query count of items sold in air conditioning category by year and display count of items sold, average count of units sold per day, and total sold on groundhog day (2/2)
- Order by year ascending

** Developer's note: we use LOWER function to do a string comparison in lower case

```
SELECT
strftime('%Y', BusinessDay.BusinessDate) AS Year,
SUM(CASE WHEN LOWER(Assignto.CategoryName) = 'air conditioning' THEN
Sells.QuantitySold ELSE 0 END) AS TotalACUnitsSold,
SUM(CASE WHEN LOWER(Assignto.CategoryName) = 'air conditioning' THEN
Sells.QuantitySold ELSE 0 END) / 365 AS AVGUnitsSoldPerDay,
SUM(CASE WHEN strftime('%m', BusinessDay.BusinessDate) = '02' AND
strftime('%d', BusinessDay.BusinessDate) = '02' THEN Sells.QuantitySold ELSE 0
END) AS GroundhogDaySales
FROM
BusinessDay
JOIN
Discount ON BusinessDay.BusinessDate = Discount.BusinessDate
JOIN
Product ON Discount.PID = Product .PID
JOIN
Assignto ON Product .PID = Assignto.PID
JOIN
Sells ON Assignto.PID = Sells.PID
GROUP BY
strftime('%Y', BusinessDay.BusinessDate)
ORDER BY
Year ASC;
```

- Generate AuditLogEntry

Developer's Note: '$EmployeeID' is acquired from the session. '$TimeStamp' is dynamically set to the current time, and '$ReportName' reflects the name of the report currently being viewed by the user.

- Insert AuditLog to AuditLogEntry

```
INSERT INTO AuditLogEntry (EmployeeID, Timestamp, ReportName)
VALUES ('$EmployeeID', '$TimeStamp', 'ReportName');
```

View Store Revenue by Year by State Report

Abstract Code:

- Check if user has been granted access to Reports

```
SELECT User.EmployeeID, COUNT(CanAccess.DistrictNumber) as CountDistrictAccess
FROM User JOIN CanAccess
ON User.EmployeeID = CanAccess.EmployeeID
GROUP BY User.employeeID
HAVING COUNT(CanAccess.DistrictNumber) = (SELECT COUNT(DistrictNumber) FROM
District);
```

- If a user is granted access, then:
  - State is selected from dropdown
  - View Store Revenue by Year by State Report
    - Query Store and calculated revenue by StoreNumber and display storeNumber, cityName, year, and total revenue of each store in selected state
    - group by year

```
SELECT Store.StoreNumber, City.CityName, strftime('%Y', Sells.DateSold) AS
Year, SUM(CASE WHEN Discount.DiscountPrice IS NOT NULL THEN
Discount.DiscountPrice * Sells.QuantitySold ELSE Product.RetailPrice *
Sells.QuantitySold END) AS total_revenue
FROM Sells JOIN Product
ON Sells.PID=Product.PID JOIN Store
ON Sells.StoreNumber=Store.StoreNumber JOIN City
```

```
ON Store.CityName=City.CityName JOIN District
ON Store.DistrictNumber=District.DistrictNumber LEFT JOIN Discount
ON Sells.PID=Discount.PID AND Sells.DateSold=Discount.BusinessDate JOIN
Holidays
ON Sells.DateSold=Holidays.BusinessDate
WHERE City.State='$State'
GROUP BY Store.StoreNumber, City.CityName, Year
ORDER BY Year ASC, total_revenue DESC;
```

- Generate AuditLogEntry

  Developer's Note: '$EmployeeID' is acquired from the session. '$TimeStamp' is dynamically set to the current time, and '$ReportName' reflects the name of the report currently being viewed by the user.

  - Insert AuditLog to AuditLogEntry

```
INSERT INTO AuditLogEntry (EmployeeID, Timestamp, ReportName)
VALUES ('$EmployeeID', '$TimeStamp', 'ReportName');
```

View District with Highest Volume for each Category Report

Abstract Code:

- Check if user has been granted access to Reports

```
SELECT User.EmployeeID, COUNT(CanAccess.DistrictNumber) as CountDistrictAccess
FROM User JOIN CanAccess
ON User.EmployeeID = CanAccess.EmployeeID
GROUP BY User.employeeID
HAVING COUNT(CanAccess.DistrictNumber) = (SELECT COUNT(DistrictNumber)
FROM District)
```

- If a user is granted access, then:

- a year and month are selected
- View District with Highest Volume for each Category Report
    - Query Category by CategoryName and display District with max quantitySold within category and quantity sold by stores
    - Group by District
    - sort by category Name ascending

```
WITH CategoryDistrictSales AS (
SELECT Assignto.CategoryName, District.DistrictNumber, Sum(CASE WHEN
strftime('%Y',Sells.DateSold) = '$Year' AND strftime('%m',Sells.DateSold)  = '$Month'
THEN Store.QuantitySold ELSE 0 END)
FROM Sells JOIN Store
ON Sells.StoreNumber = Store.StoreNumber JOIN District
ON Store.DistrictNumber = District.DistrictNumber JOIN Product
ON Sells.PID = Product.PID JOIN Assignto
ON Product.PID = Assignto.PID
GROUP BY Assignto.CategoryName, District.DistrictNumber),
MaxCategorySales AS
(
SELECT CategoryName, MAX(TotalUnitsSold) AS MaxUnitsSold
FROM CategoryDistrictSales
GROUP BY CategoryName
)
Select CategoryDistrictSales.CategoryName, CategoryDistrictSales.DistrictNumber,
CategoryDistrictSales.TotalUnitsSold AS MaxUnitsSold
FROM CategoryDistrictSales JOIN MaxCategorySales
ON CategoryDistrictSales.CategoryName = MaxCategorySales.CategoryName AND
CategoryDistrictSales.TotalUnitsSold = MaxCategorySales.MaxUnitsSold
ORDER BY CategoryDistrictSales.CategoryName ASC;
```

- Select District, category, year/month for more details
    - from selected district, category, year/month, display IDs, states, and cities of all stores

      ** Developer's note: We assume that this is what a user would see from our final Web UI application. A user can see drill-down details for a selected record by clicking the Details button here.

	CategoryName	DistrictNumber	MaxUnitsSold	As Of (YYYY-MM)
Details	Car	3	10	2024-05
Details	Office facility	2	5	2024-05

** Developer's note: A new detailed report should get a list of records to display all stores in a given district where a product of this given category was sold. We assume that our final UI would look like this.

CategoryName: Car, DistrictNumber: 3, As Of: 2024-05

Store #	Address	City	State	
1	44 Wall Street	New York	NY	
2	1 Broadway	New York	NY	

```
SELECT Store.StoreNumber, City.CityName, City.State
FROM Store, City, Sells, Assignto
WHERE
Store.DistrictNumber = '$districtNumber' AND
Store.StoreNumber = Sells.StoreNumber AND
Store.CityName = City.CityName AND
Sells.PID = Assignto.PID AND
Assignto.CategoryName = '$categoryName' AND
strftime('%Y',Sells.DateSold) = '$selectedYear' AND
strftime('%m',Sells.DateSold) = '$selectedMonth'
```

- Generate AuditLogEntry

Developer's Note: '$EmployeeID' is acquired from the session. '$TimeStamp' is dynamically set to the current time, and '$ReportName' reflects the name of the report currently being viewed by the user.

  - Insert AuditLog to AuditLogEntry

```
INSERT INTO AuditLogEntry (EmployeeID, Timestamp, ReportName)
VALUES ('$EmployeeID', '$TimeStamp', 'ReportName');
```

View Revenue by Population Report

Abstract Code:

- Check if user has been granted access to Reports

```
SELECT User.EmployeeID, COUNT(CanAccess.DistrictNumber) as CountDistrictAccess
FROM User JOIN CanAccess
ON User.EmployeeID = CanAccess.EmployeeID
GROUP BY User.employeeID
HAVING COUNT(CanAccess.DistrictNumber) = (SELECT COUNT(DistrictNumber) FROM
District)
```

- If a user is granted access, then:
    - Find total Revenue by Year and size of cities and display revenue by city size by year

```
SELECT
strftime('%Y', BusinessDay.BusinessDate) AS Year,
CASE
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 'Extra Large'
END AS CitySize,
SUM(Sells.QuantitySold * Product.RetailPrice) AS TotalRevenue
FROM Sells JOIN Store
ON Sells.StoreNumber = Store.StoreNumber JOIN City
ON Store.CityName = City.CityName JOIN Product
ON Sells.PID = Product.PID JOIN BusinessDay
ON Sells.DateSold = BusinessDay.BusinessDate
GROUP BY Year, CitySize
ORDER BY Year ASC, CitySize ASC;
```

- Generate AuditLogEntry

  Developer's Note: '$EmployeeID' is acquired from the session. '$TimeStamp' is dynamically set to the current time, and '$ReportName' reflects the name of the report currently being viewed by the user.

  - Insert AuditLog to AuditLogEntry

```
INSERT INTO AuditLogEntry (EmployeeID, Timestamp, ReportName)
VALUES ('$EmployeeID', '$TimeStamp', 'ReportName');
```

View Audit Logs

Abstract Code

- Check if user has been granted to access to Audit Logs

```
SELECT AccessToAuditLog
From User
WHERE EmployeeID = '$session.employeeID';
```

- If a user is granted to access(above statement outputs TRUE), then:
    - View Top 100 most recent audit logs

```
SELECT TOP 100 AuditLogEntry.ReportName, User.EmployeeID,
User.FirstName, User.LastName, AuditLogEntry.TimeStamp
FROM AuditLogEntry
JOIN User ON AuditLogEntry.EmployeeID = User.EmployeeID
ORDER BY AuditLogEntry.TimeStamp DESC;
```

    - Choose an ***Audit Log View*** button to view
    - A selected log can be printed or exported to other file formats

- Else:
    - Stay in **Main Menu** form