Database Project

Abstract Code w/ SQL

Table of Contents:

BuzzBuy Data Warehouse Task Decomposition with Abstract Code

- Login
- View Main Menu Screen
- View Audit Log
- Update Audit Log
- View Holidays
- Add/Delete Holidays
- Warehouse Reports
  - General Reports
    - Report 1 - Manufacturer's Product Report
    - Report 2 - Category Report
  - District Reports
    - Report 3 - Actual vs Predicted Revenue for GPS units
    - Report 4 - Air Conditioners on Groundhog Day
  - Corporate Reports
    - Report 5 - Store Revenue by Year by State
    - Report 6 - District with Highest Volume for Each Category
    - Report 6 - District with Highest Volume Drill Down
    - Report 7 - Revenue by Population

Login

Abstract Code:

- User enters employee ID('$employeeID') and password('$Password')
- If data validation is successful for both employee ID and password input fields, then:
    - When ***Enter*** button is clicked:

```
SELECT CONCAT(SUBSTRING(SSN,5,4), `-`,
UPPER(LEFT(LName,1)), LOWER(SUBSTRING(LName,2,LENGTH(LName)))) AS
password
FROM employee WHERE EmployeeID = `$employeeID`;
```

- ➢ If User record is found but user.password != `$password`:
    - Returns an error message and go back to **Login** form
- ➢ Else:
    - Store login information as session variable `$UserID`.
    - Go to **View Main Menu Screen**

- Else employeeID and password input fields are invalid, display **Login** form, with error message.

View Main Menu Screen

Abstract Code:

- User successfully logs in
- Display welcome message "Welcome,FName LName"

```
SELECT CONCAT(FName, " ", LName) AS FullName
FROM employee
WHERE EmployeeID = `$employeeID`;
```

- Calculate and display statistics (count of store, city, district, manufacturer, product, category, and holiday)

```
SELECT COUNT(*) AS store_count  FROM store;

SELECT COUNT(*) AS city_count  FROM city;

SELECT COUNT(*) AS district_count FROM district;

SELECT COUNT( DISTINCT Manufacturer) AS manu_count
FROM product;

SELECT COUNT(*) AS product_Count FROM product;

SELECT COUNT( DISTINCT CategoryName) AS category_count
FROM product_category;
```

```
SELECT COUNT(*) AS holiday_count FROM holiday;
```

- Show available reports based on assigned district and AuditLogFlag

```
SELECT DistrictNumber  FROM assigned
WHERE EmployeeID = `$employeeID`;

SELECT AuditLogFlag   FROM employee
WHERE EmployeeID = `$employeeID`;
```

  - When user assigned to 1 or more districts
    - Display General Reports
      - ***Report 1 - Manufacturer's Product Report***
      - ***Report 2 - Category Report***
    - Display District Reports for assigned districts
      - ***Report 3 - Actual versus Predicted Revenue for GPS units***
      - ***Report 4 - Air Conditioners on Groundhog Day?***
    - Display ***View Holidays***
  - When user assigned to all districts, also display
    - Corporate Reports
      - ***Report 5 - Store Revenue by Year by State***
      - ***Report 6 - District with Highest Volume for each Category***
      - ***Report 7 - Revenue by Population***
    - Display ***Add/Delete Holidays***
  - When USER AuditLogFlag == 1
    - Display ***Audit Log Report***
- Upon:
  - Click ***Report 1 - Manufacturer's Product Report*** - Jump to **Report 1 - Manufacturer's Product Report** Task.
  - Click ***Report 2 - Category Report*** - Jump to **Report 2 - Category Report** Task.
  - Click ***Report 3 - Actual versus Predicted Revenue for GPS units*** - Jump to **Report 3 - Actual versus Predicted Revenue for GPS units** Task.
  - Click ***Report 4 - Air Conditioners on Groundhog Day?*** - Jump to **Report 4 - Air Conditioners on Groundhog Day?** Task.
  - Click ***Report 5 - Store Revenue by Year by State*** - Jump to **Report 5 - Store Revenue by Year by State** Task.
  - Click ***Report 6 - District with Highest Volume for each Category*** - Jump to **Report 6 - District with Highest Volume for each Category** Task.
  - Click ***Report 7 - Revenue by Population*** - Jump to **Report 7 - Revenue by Population** Task.
  - Click ***Audit Log*** - Jump to **View Audit Log** Task.
  - Click ***View Holidays*** - Jump to **View Holidays** Task.
  - Click ***Add/Delete Holidays*** - Jump to **Add/Delete Holidays** Task.

View Audit Log

Abstract Code

- User clicks on ***View Audit Log*** button from **Main Menu:**
- Run the **View Audit Log** task: query for the most recent 100 audit log records
    - Display employeeID, LName, FNmae, DateAndTime, ReportViewed

```
SELECT DateAndTime, EmployeeID, CONCAT(LName, ", ",  FName) AS FullName,
ReportViewed
FROM employee NATURAL JOIN audit_log
ORDER BY  DateAndTime DESC
LIMIT 100;
```

- Upon:
    - Click ***Main Menu*** button - Jump to **View Main Menu Screen** task

Update Audit Log

Abstract Code

- User clicks on ***View Report*** button from **Main Menu:**
- Run the **Update Audit Log** task: query for information about the user where ('$UserID') is the employeeID of the current user using the system from the HTTP Session/Cookie
    - Record DateAndTime and ReportViewed

```
INSERT INTO audit_log (DateAndTime, ReportViewed, EmployeeID)
          VALUES(CURRENT_TIMESTAMP(), $ReportViewed, $UserID);
```

- Upon:
    - Click ***Main Menu*** button - Jump to **View Main Menu Screen** task

View Holidays

Abstract Code

- Upon user click on ***View Holidays*** button from Main Menu
    - Run the **View Holidays** task; query for all holidays and present table.

```
SELECT HDate AS HolidayDate, HName AS HolidayName, AddedByEID
FROM holiday
ORDER BY HDate DESC;
```

- Upon:
    - Click ***Main Menu*** - Jump to **view Main Menu Screen**

Add/Delete Holidays

Abstract Code

- Users assigned to all districts clicked on ***Add/Delete Holidays*** button from Main Menu
- Run **Add/Delete Holidays** task; query for all holidays and display table, ***Delete*** button and ***Add*** buttons are shown below table.

```
SELECT HDate, HName, AddedByEID
FROM holiday;
```

- Upon:
    - Click ***Delete*** button- Run **Delete Holiday** task; Opens new window with field to enter a delete date ($DeleteDate).
    - Upon:
        - User entering delete date, and clicking the ***Ok*** button:
        - Checks for date data type in delete date field.
            - If data type check fails:
                - Show “Data type error - try again” message
                - Rerun **Delete Holiday** task.
            - Else:
                - Delete holiday from database

```
DELETE FROM holiday
WHERE Hdate = '$DeleteDate';
```

- Rerun **Add/Delete Holidays** task.

- Upon:
    - Click ***Add*** button- Run **Add Holiday** task: Opens new window with fields to enter a holiday name *('*$HolidayName*')* then holiday date ('$HolidayDate').
    - Upon:
        - User entering *name*, *date*, and clicking the ***Ok*** button:
        - Checks data type for string in HName field and date data type in HDate field.
            - If either data type check fails:
                - Show “Data type error - try again” message
                - Rerun **Add Holiday** task
            - Else:

- Read the database to check whether a holiday already exists on the entered *date*.

```
SELECT  HDate, HName, AddedByEID FROM holiday
WHERE Hdate = $HolidayDate;
```

- If Holiday already exists:
    - Show "Holiday already exists" message
    - Rerun **Add Holiday** task
- Else:
    - Add new holiday to the database.

```
INSERT INTO holiday(HDate, HName, AddedByEID)
VALUES ($HolidayDate, $HolidayName, $UserID);
```

- Rerun **Add/Delete Holidays** task.
- Upon:
    - ➢ Click ***Main Menu*** - Jump to **view Main Menu Screen**

Warehouse Reports

General Reports

Report 1 - Manufacturer's Product Report

Abstract Code

- User clicks on ***Report 1 - Manufacturer's Product Report*** button from **<u>Main Menu</u>**
    - Run **Update Audit Log** Task
- Run the Report 1 task; query for information about manufacturer's and their products where ('$manufacturer') is the name of the product manufacturer
    - Count the number of products offered by the manufacturer
    - Calculate the average retail price of the all the manufacturer's products
    - Calculate the minimum retail price of the all the manufacturer's products
    - Calculate the maximum retail price of the all the manufacturer's products
    - Sort results by average price with the highest average price, display only top 100 results

```
SELECT Manufacturer, COUNT(DISTINCT PID) AS NumOfProducts,
            AVG(RetailPrice)  AS AvgPrice,
            MIN(RetailPrice) AS MinPrice,
            MAX(RetailPrice) AS MaxPrice
FROM product
GROUP BY Manufacturer
ORDER BY  AvgPrice  DESC
LIMIT 100;
```

- Upon:
    - Click ***Manufacturer Name*** - **Report 1 - Manufacturer's Report**
        - Query for information about each manufacturer's products based on the $Manufacturer the use chooses

```
SELECT PR.PID,PName, RetailPrice,
            GROUP_CONCAT(CategoryName SEPARATOR ", ") AS Category
FROM product AS PR
JOIN product_category AS PC ON PR.PID = PC.PID
WHERE Manufacturer = $Manufacturer
GROUP BY PR.PID
ORDER BY RetailPrice DESC;
```

        - Display manufacturer name and summary information from **Report 1 - Manufacturer's Report**
        - Display list of products including product ID, name, category (or categories), and price
    - Click ***Main Menu*** - Jump to **View Main Menu Screen**

Report 2 - Category Report

Task Decomposition

Abstract Code

- User clicks on ***Report 2 - Category Report*** button from **Main Menu**
    - Run **Update Audit Log** Task
- Run the **Report 2 - Category Report** task; query for information about each category.
    - Return the category CName
    - Calculate Total Number Of Products
    - Calculate Total Number Of Manufacturers
    - Calculate Average Price
    - Display results table sorted by category name in ascending order

```
SELECT CategoryName, COUNT(DISTINCT PID) AS NumOfProd,
            COUNT(DISTINCT Manufacturer) AS NumOfManu, AVG(RetailPrice) AS
AvgPrice
FROM product NATURAL JOIN product_category
GROUP BY CategoryName
ORDER BY CategoryName;
```

- Upon:
    - Click ***Main Menu*** - Jump to **View Main Menu Screen**

District Reports

Report 3 - Actual vs Predicted Revenue for GPS units

Abstract Code

- User clicks on ***Report 3 - Actual vs Predicted Revenue for GPS units*** button from the **Main Menu**
    - Run **Update Audit Log** Task
- Run the Report 3 task; query for information about the actual vs predicted revenue for GPS units; Display screen for **Report 3 - Actual vs Predicted Revenue for GPS Units**
    - Find all products assigned to the GPS category using the CName == “GPS”. For each product (assigned using the GROUP BY PID):
        - ➢ Query the PID, PName, RetailPrice; Display
        - ➢ Query all total number of units sold with the current product:
            - $total_units_sold is the count of this per product; Display this total
        - ➢ Query total number of units sold at a discount:
            - $units_sold_at_discount_price; Display this total
        - ➢ Query total number of units sold at the retail price
            - $units_sold_at_retail_price; Display this total
        - ➢ Query actual revenue:
            - $total_actual_revenue: Multiply the quantity of units sold at discount_price to the price + multiply the quantity of units sold at retail price to the price
        - ➢ Query predicted revenue:
            - $total_predicted_revenue: Find all transactions that are sold at a discount, calculate the ((quantity * 0.75) * retail price) and sum together to get this value; Display this total
        - ➢ Query difference of actual and predicted revenues if the absolute value is larger than 200:
            - $actual_vs_predicted_rev_difference; display
            - Calculation:
                - If absolute value of ($total_actual_revenue - $total_predicted_revenue) < 200, set value to NULL
                - Else: display ($total_actual_revenue - $total_predicted_revenue)
    - Sort the predicted revenues by descending order

```
SELECT
  -- Query the PID, PName, RetailPrice
  P.PID,
  P.PName,
  P.RetailPrice,
  -- Query all total number of units sold with the current product
  COALESCE(SUM(S.Quantity), 0) AS Total_Units_Sold,
  -- Query total number of units sold at a discount
  (
      SELECT COALESCE(SUM(S2.Quantity), 0)
      FROM sale S2
      JOIN Discount D ON S2.PID = D.PID
                  AND S2.SaleDate = D.DiscountedDate
      WHERE P.PID = S2.PID
  ) AS Units_Sold_At_Discount_Price,

  -- Query total number of units sold at the retail price
  (
      SELECT COALESCE(SUM(S3.Quantity), 0)
      FROM sale S3
      LEFT JOIN discount D2 ON S3.PID = D2.PID
                      AND S3.SaleDate = D2.DiscountedDate
      WHERE P.PID = S3.PID AND D2.DiscountedDate IS NULL
  ) AS Units_Sold_At_Retail_Price,

  -- Query actual revenue
  (
      (SELECT COALESCE(SUM(S4.Quantity * D3.DiscountPrice), 0)
      FROM sale S4
      JOIN discount D3 ON S4.PID = D3.PID
                  AND S4.SaleDate = D3.DiscountedDate
      WHERE P.PID = S4.PID
      ) + (
      SELECT COALESCE(SUM(S5.Quantity * P.RetailPrice), 0)
      FROM sale S5
      LEFT JOIN discount D4 ON S5.PID = D4.PID
                      AND S5.SaleDate = D4.DiscountedDate
      WHERE P.PID = S5.PID AND D4.DiscountedDate IS NULL
      )
  ) AS Total_Actual_Revenue,

  -- Query predicted revenue
  (
      (SELECT COALESCE(SUM(S6.Quantity * 0.75 * P.RetailPrice), 0)
      FROM sale S6
      JOIN discount D5 ON S6.PID = D5.PID
                  AND S6.SaleDate = D5.DiscountedDate
      WHERE P.PID = S6.PID
      ) + (
      SELECT COALESCE(SUM(S7.Quantity * P.RetailPrice), 0)
      FROM sale S7
      LEFT JOIN discount D6 ON S7.PID = D6.PID
                      AND S7.SaleDate = D6.DiscountedDate
      WHERE P.PID = S7.PID AND D6.DiscountedDate IS NULL
      )
  ) AS Total_Predicted_Revenue,

  -- Query difference of actual and predicted revenues if the absolute value is larger
than 200
  ((SELECT COALESCE(SUM(S8.Quantity * D7.DiscountPrice), 0)
      FROM sale S8
      JOIN discount D7 ON S8.PID = D7.PID
                AND S8.SaleDate = D7.DiscountedDate
      WHERE P.PID = S8.PID
      ) + (
      SELECT COALESCE(SUM(S9.Quantity * P.RetailPrice), 0)
      FROM sale S9
      LEFT JOIN discount D8 ON S9.PID = D8.PID
                    AND S9.SaleDate = D8.DiscountedDate
      WHERE P.PID = S9.PID AND D8.DiscountedDate IS NULL
      )) -
      ((SELECT COALESCE(SUM(S10.Quantity * 0.75 * P.RetailPrice), 0)
      FROM sale S10
      JOIN discount D9 ON S10.PID = D9.PID
                      AND S10.SaleDate = D9.DiscountedDate
      WHERE P.PID = S10.PID
      ) +
      (SELECT COALESCE(SUM(S11.Quantity * P.RetailPrice), 0)
      FROM sale S11
      LEFT JOIN discount D10 ON S11.PID = D10.PID
                      AND S11.SaleDate = D10.DiscountedDate
      WHERE P.PID = S11.PID AND D10.DiscountedDate IS NULL
      )) AS Actual_Vs_Predicted_Rev_Difference

FROM sale S
LEFT JOIN product P ON P.PID = S.PID
LEFT JOIN product_category PC ON P.PID = PC.PID
WHERE PC.CategoryName = 'GPS'
GROUP BY P.PID
HAVING Actual_Vs_Predicted_Rev_Difference > 200 OR
Actual_Vs_Predicted_Rev_Difference < -200
ORDER BY Actual_Vs_Predicted_Rev_Difference DESC;
```

- Upon:
    - Click ***Main Menu*** - Jump to **view Main Menu Screen**

Report 4 - Air Conditioners on Groundhog Day

Abstract Code

- User clicks on ***Report 4*** button from the **<u>Main Menu</u>**
    - Run **Update Audit Log** Task
- **Run the Report 4 task**; query for information about air conditioners on groundhog day; Display screen for **<u>Report 4 - Air Conditioners on Groundhog Day</u>**
    - Find all Products assigned to the Category using CName == “Air Conditioning”
        - ➢ For all products, find all transactions sold
            - From the Transaction SaleDates, find all the years. For each year:
                - Display the year
                - Sum the total number of items sold that year

        - Calculate average units sold per day by dividing this sum by 365; Display as the (‘AverageNumberofUnitsSoldPerDay’)
    - Find the holiday with HName == “Groundhog Day” and match with the transactions that have this holiday
        - Sum the total number of transactions with this holiday; Display as the (‘TotalNumberofUnitsSoldOnGroundhogDay’)
- ➢ Sort the report on the year in ascending order

```
SELECT
   YEAR(S.SaleDate) AS Year,
   COALESCE(SUM(S.Quantity), 0) AS Total_Quantity,
   (COALESCE(SUM(S.Quantity), 0) * 1. / 365) AS Avg_Units_Sold_Per_Day,
   (
      SELECT COALESCE(SUM(S2.Quantity), 0)
      FROM sale S2
      WHERE DATE_FORMAT(SaleDate, '%m%d') = '0202'
      AND YEAR(S2.SaleDate) = YEAR(S.SaleDate) ) AS
Units_Sold_On_Groundhog_Day
FROM sale S
LEFT JOIN product P ON P.PID = S.PID
LEFT JOIN product_category PC ON P.PID = PC.PID
WHERE PC.CategoryName = 'Air Conditioning'
GROUP BY YEAR(S.SaleDate)
ORDER BY YEAR(S.SaleDate);
```

- Upon:
    - Click ***Main Menu*** - Jump to **view Main Menu Screen**

Corporate Reports

Report 5 - Store Revenue by Year by State

Abstract Code

- User clicks on ***Report 5*** button from **Main Menu**
    - Run **Update Audit Log** Task;
    - Query City table for all states and displays results in a drop-down menu.

```
SELECT StateName FROM city;
```

- Upon:
    - Selecting a State ('$State) from the drop-down menu.
    - Run the **Report 5** task; query all transactions sold in Stores grouped by Year where ('$State) is the selected name of the state to which the Stores belong.
    - For each transaction sold in stores located in the selected state:
        - ➢ Display the StoreNumber and CityName.

            - For each product PID in each transaction:
                - If SaleDate equals DiscountedDate:
                    - Display DiscountedPrice
                - Otherwise display RetailPrice
        - For each group:
            - Calculate **Total Revenue**
        - Display report sorted first by year in ascending order and then by revenue in descending order.

```
SELECT StoreNumber, CityName, YEAR(SaleDate) AS Year,
          SUM(SalePrice * Quantity) AS Revenue
FROM sale NATURAL JOIN store NATURAL JOIN city
WHERE StateName = $State
GROUP BY StoreNumber, Year, StoreAddress, CityName
ORDER BY Year ASC, Revenue DESC;
```

- Upon:
    - Click ***Main Menu*** - Jump to **view Main Menu Screen**

Report 6 - District with Highest Volume for each Category

Abstract Code

- User clicks on ***Report 6*** button from **<u>Main Menu</u>**
- Run the **Update Audit Log** task: query for information about the user where ("$UserID") is the employeeID of the current user using the system from the HTTP Session/ Cookie and record DateAndTime and ReportViewed

```
INSERT INTO audit_log (DateAndTime, ReportViewed, EmployeeID)
          VALUES(CURRENT_TIMESTAMP(), $ReportViewed, $UserID);
```

- Display **<u>Report 6</u>** and populate the Year/ Month drop-down menu:

```
SELECT DISTINCT DATE_FORMAT(SaleDate, '%Y%m') AS YearMonth
FROM sale;
```

- User will select a Year/ Month from the drop-down menu
- Upon:
    - User clicks on ***View Report*** button:
        - Clear any previously written lines
        - Read the selected YearMonth($YearMonth) from the drop-down menu
        - With a hyperlink associated with each category and district, display results of the following SQL:

```
SELECT Category, DistrictNumber, UnitsSold
FROM
      (SELECT unitsCategory AS Category, MAX(units) AS UnitsSold
      FROM
            (SELECT CAT.CategoryName AS unitsCategory, ST.DistrictNumber,
                        SUM(Quantity) AS units
            FROM sale SA
            JOIN product P
            ON SA.PID = P.PID
            JOIN store ST
            ON SA.StoreNumber = ST.StoreNumber
            JOIN product_category PC
            ON SA.PID = PC.PID
            JOIN category CAT
            ON CAT.CategoryName = PC.CategoryName
            WHERE DATE_FORMAT(SaleDate, '%Y-%m') = '" . $selected_date . "'
            GROUP BY CAT.CategoryName, ST.DistrictNumber) AS Q1
GROUP BY unitsCategory) AS Q2
JOIN
        (SELECT CAT.CategoryName AS DisCategory, ST.DistrictNumber AS
                    DistrictNumber, SUM(Quantity) AS DisUnits
        FROM sale SA
        JOIN product P
        ON SA.PID = P.PID
        JOIN store ST
        ON SA.StoreNumber = ST.StoreNumber
        JOIN product_category PC
        ON SA.PID = PC.PID
        JOIN category CAT
        ON CAT.CategoryName = PC.CategoryName
        WHERE DATE_FORMAT(SaleDate, '%Y-%m') = '" . $selected_date . "'
        GROUP BY CAT.CategoryName, ST.DistrictNumber) AS Q3
ON Category = DisCategory AND UnitsSold = DisUnits
ORDER BY Category;"
```

```
WITH units_sold AS (
   SELECT CategoryName, DistrictNumber, SUM(Quantity) AS UnitsSold
   FROM sale NATURAL JOIN product_category NATURAL JOIN store
   WHERE DATE_FORMAT(SaleDate, '%Y%m') = '$YearMonth'
   GROUP BY CategoryName, DistrictNumber
),
ranked_units_sold AS (
   SELECT
      CategoryName,
      DistrictNumber,
      UnitsSold,
      ROW_NUMBER() OVER (PARTITION BY CategoryName ORDER BY UnitsSold
DESC) AS rn
   FROM units_sold
)
SELECT
   CategoryName,
   DistrictNumber,
   UnitsSold
FROM ranked_units_sold
WHERE rn = 1
ORDER BY CategoryName;
```

- Upon:
  - User clicks on ***District*** button
    - Clear the drill-down area provided at the bottom of the **<u>Report 6</u>**
    - Determine the Category($category) and District($district)
      associated with the clicked hyperlink
    - Display the Category($category), YearMonth($YearMonth), and District($district) in the header area
    - Display results of the following SQL in the drill-down area:

```
SELECT StoreNumber, ST.StateName, ST.CityName
FROM store ST, city C
WHERE ST.CityName = C.CityName AND ST.StateName = C.StateName
AND DistrictNumber = 5
ORDER BY CAST(StoreNumber AS INT) ASC;
```

- User clicks on ***Main Menu*** button
  - Run the **View Main Menu Screen** task

Report 7 - Revenue by Population
Abstract Code

- User clicks on ***Report 7*** from **Main Menu**
- Run the **Update Audit Log** task: query for information about the user where ("$UserID") is the employeeID of the current user using the system from the HTTP Session/ Cookie and record DateAndTime and ReportViewed

```
INSERT INTO audit_log (DateAndTime, ReportViewed, EmployeeID)
        VALUES(CURRENT_TIMESTAMP(), $ReportViewed, $UserID);
```

- Display **Report 7**

```
SELECT YEAR(SaleDate) AS TheYear,
        CASE WHEN PopulationSize < 3700000 THEN "Small"
              WHEN PopulationSize >= 3700000 AND  PopulationSize < 6700000
THEN "Medium"
              WHEN PopulationSize >= 6700000 AND  PopulationSize < 9000000
THEN "Large"
              WHEN PopulationSize >= 9000000 THEN "ExtraLarge"
         END AS CitySize,
         AVG(SalePrice * Quantity) AS AvgRevenue
FROM sale SA, store ST, city C
WHERE SA.StoreNumber = ST.StoreNumber AND ST.CityName = C.CityName
        AND ST.StateName = C.StateName
GROUP BY TheYear, CitySize
ORDER BY TheYear, PopulationSize;
```

- Upon:
  - User clicks on ***Main Menu*** button
    - Run the **View Main Menu Screen** task