**School of Computing**

**ST1501 Data Engineering CA1**

**AY2023/2024 Semester 1**

|  |  |
| --- | --- |
| **Section A - Database Design (Entity Relationship Diagram)** | 30% |
|  | | |

|  |  |
| --- | --- |
| **Section B - Database Design (Database Diagram)**   * The database supports the described business scenario. * The chosen table names, field names and attributes are descriptive. * The database is normalized. | 10% |
|  | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Section C - Database Creation**   * Attach an SQL file to include create table statememts including the primary key and foreign key definition. * Enter the details of each table in the template. | | | | 10% |
| **List all the relations, PK, FK and all non key attributes for the database relation diagram. Attach an SQL file to show the create table SQL script to implementthe database design, including the primary key and foreign key definition.** | | | | | | | |
| **No** | **Relation Name** | **Primary Key** | **Non-key Attribute(s)** | | | **Foreign Key(s)** | **Referenced Relation** |
| 1 | Employee | EmpID | Name | | | StoreID | Store(StoreID) |
| SSN | | | ManagerID | Employee(EmpID) |
| Phone | | |  |  |
| Address | | |
| Email | | |
| Password | | |
| Date\_Start | | |
| Date\_End | | |
| Date\_Hired | | |
| HourlyPay | | |
| Yearly Pay | | |
| ChngPwdDate | | |
| 2 | Dependent | EmpID | Phone | | | EmpID | Employee(EmpID) |
| Name | Relation | | |  |
| 3 | Store | StoreID | Address | | | ManagerID | Employee(EmpID) |
| 4 | Item | ItemID | Brand | | |  |  |
| Tax | | |
| Price | | |
| Shape | | |
| Size | | |
| Cost | | |
| Weight | | |
| Description | | |
| UPC | | |
| 5 | Checkout | CustID | DateOfPurchase | | | CustID | Customer(CustID) |
| EmpID | EmpID | Employee(EmpID) |
| 6 | Inventory | StoreID | Quantity | | | StoreID | Store(StoreID) |
| ItemID | ItemID | Item(ItemID) |
| 7 | Customer | CustID | Name | | |  |  |
| Phone | | |
| Email | | |
| Date\_Joined | | |
| 8 | CheckOutItems | CheckOutID | Quantity | | | CheckOutID | Checkout(CheckOutID) |
| ItemID | ItemID | Item(ItemID) |

|  |
| --- |
| **Section D - The Query Statements 35%** |
| Query 1  SELECT C.Name as "CustName", S.StoreID as "Store\_ID", S.Address as "Store Address",  M.Name as "Manager of The Store", O.DateOfPurchase as "Transaction Date", I.Brand, I.Description,  CO.Quantity as "Quantity Purchased", E.Name as "Served By"  FROM CheckOut O  JOIN Customer C ON O.CustID = C.CustID  JOIN CheckOutItems CO ON O.CheckOutID = CO.CheckOutID  JOIN Item I ON CO.ItemID = I.ItemID  JOIN Employee E ON O.EmpID = E.EmpID  JOIN Store S ON E.StoreID = S.StoreID  JOIN Employee M ON S.ManagerID = M.EmpID; |
| Query 1 results |
| Query 2  SELECT S.ManagerID as "Store Manager ID", E.Name as "Name of Manager", S.StoreID as "Store Managed", I.ItemID as "Item",  Iv.Quantity as "Quantity on Inventory"  FROM Inventory Iv  JOIN Item I ON Iv.ItemID = I.ItemID  JOIN Store S ON Iv.StoreID = S.StoreID  JOIN Employee E ON E.EmpID = S.ManagerID; |
| Query 2 results |
| Query 3  -- include those that never buy anything so i do left and right join  SELECT X.Name AS "Customers Who Bought <= 2 On Any Single Transaction"  FROM (  SELECT C.Name, SUM(CI.Quantity) AS Qty  FROM Customer C  LEFT OUTER JOIN CheckOut O ON O.CustID = C.CustID -- include those who nvr buy anything  LEFT OUTER JOIN CheckOutItems CI ON CI.CheckOutID = O.CheckOutID  GROUP BY C.Name, O.CheckOutID  ) AS X  GROUP BY X.Name  HAVING MAX(ISNULL(X.Qty, 0)) <= 2; -- find max items quantity on any trans |
| Query 3 results |
| Query 4  SELECT I.[ItemID] as "Item\_ID", I.Description,  SUM(I.Price \* Inv.Quantity) as 'Retail',  SUM(I.Cost \* Inv.Quantity) as 'WholeSale'  FROM Item I  JOIN Inventory Inv ON Inv.ItemID = I.ItemID  JOIN Store S ON S.StoreID = Inv.StoreID  GROUP BY I.ItemID, I.Description  HAVING COUNT(Inv.StoreID) >= 2; |
| Query 4 results |
| Query 5  SELECT E.Name as "Name of Employee", E.EmpID as "Employee ID",  M.Name as "Name of Manager", M.EmpID as "Manager ID"  FROM Employee E  JOIN Employee M ON E.ManagerID = M.EmpID; |
| Query 5 results |
| Query 6  SELECT DISTINCT M.Name as "Name of Manager", B.EmpID as "Boss ID", M.EmpID as "Manager ID",  B.Name as "Name of Boss", M.StoreID as "Store ID", S.Address  FROM Employee E  JOIN Employee M ON E.ManagerID = M.EmpID  JOIN Employee B ON M.ManagerID = B.EmpID  JOIN Store S ON M.StoreID = S.StoreID  WHERE M.StoreID = B.StoreID; |
| Query 6 results |