

Team Error 404s

MIS 531: Enterprise Data Management

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Arnav Singh, Pratiksha Rao, Amshitha Nair, Karan Salot, Kevin Selby

# Contents

[Chapter 1: Requirement Analysis 4](#_Toc184866880)

[Chapter 2: Conceptual Schema 6](#_Toc184866881)

[2A. ER DIAGRAM: 6](#_Toc184866882)

[2B. Data Dictionary (Conceptual/ER) 6](#_Toc184866883)

[Chapter 3: Relational Schema 22](#_Toc184866884)

[3A. ER to Relational Normalized Tables: 22](#_Toc184866885)

[3B. DDL Appendix: 48](#_Toc184866886)

[Chapter 4: Queries 58](#_Toc184866887)

[1. Sales Specialist Performance Report​ 58](#_Toc184866888)

[2. Customer Claim History​ 58](#_Toc184866889)

[3. Ranking Claim Specialists by Approval Amount and Claim Count 59](#_Toc184866890)

[4. Claim approval trend analysis over the past 90 days 60](#_Toc184866891)

[5. Customer Claim Analysis by State with Ranking and Subtotals 61](#_Toc184866892)

[6. Query to Calculate Total Claim Amounts with ROLLUP for Hierarchical Summary by Dealer 62](#_Toc184866893)

[7. Find Customers with Above Average number of Claims 63](#_Toc184866894)

[8. Find Customers Who Have Never Subscribed to the Most Popular Plan 63](#_Toc184866895)

[9. Long Processing Time Claims Analysis 64](#_Toc184866896)

[10. Identify Top 3 Resellers by Sales in Each City 64](#_Toc184866897)

[Chapter 5: Procedures And Triggers 66](#_Toc184866898)

[1. Procedure 1 66](#_Toc184866899)

[2. Trigger 1 : Contract End Date Calculation 67](#_Toc184866900)

[3. Trigger 2: Claim Validation Rules 68](#_Toc184866901)

[Chapter 6: Frontend Overview 71](#_Toc184866902)

[Chapter 7: Implementation Plan 73](#_Toc184866903)

[Step 1: Preparation and Requirement Analysis 73](#_Toc184866904)

[Step 2: Environment Setup 73](#_Toc184866905)

[Step 3: Schema Creation 73](#_Toc184866906)

[Step 4: Data Migration 73](#_Toc184866907)

[Step 5: Business Logic Implementation 74](#_Toc184866908)

[Step 6: Testing and Optimization 74](#_Toc184866909)

[Step 7: Deployment 75](#_Toc184866910)

[Step 8: Maintenance and Monitoring 75](#_Toc184866911)

[References 76](#_Toc184866912)

[Appendix A: Lessons Learned 77](#_Toc184866913)

# Chapter 1: Requirement Analysis

MHHC Enterprises operates within the warranty and asset servicing industry, managing a complex network of interconnected entities such as customers, resellers, agents, servicers, and employees. To streamline operations and enhance service quality, a robust database system is required.

MHHC works with two types of resellers: Agents and Enterprise Resellers. Each reseller is identified by Reseller ID, Name, Contact Person, Email, Phone, Address, and Region. Additional attributes for agents include Agent ID, Location, and Phone Number, while enterprise resellers require data on Sales Count, Account Status, and Registration Date. Resellers acquire policies based on reseller catalogs, tracked with Reseller Catalog ID, Policy ID, Retail Price, and Additional Coverage Notes.

Customers are categorized as Individual Customers and Enterprise Businesses. For Individual Customers, the database stores Customer ID, First Name, Last Name, Phone Number, Email, Address, Number of Contracts, and Total Claims. Enterprise Businesses have additional attributes such as Name, ID, Number of Sales, and Transactions with Enterprise Customers. Enterprise Customers are further detailed with ID, Name, Email, and Contracts Assigned.

Claims are filed by customers and linked to assets. Each claim is identified by Claim ID, Loss Date, and Problem Description. Claims undergo review by claims specialists and are associated with Claim Approval Reports, which include Approval ID, Status, Date, Amount, and Review Date. Assets are uniquely identified by Asset ID and include Brand, Model, Serial Number, Condition, Retail Value, MFG Labor End, and MFG Part End. Each asset can be linked to one claim and one repair.

Repairs are executed by servicers and are tracked using Repair ID, Repair Date, Description, Resolution, Claim Amount, Labor Costs, Parts, and Shipping Expenses. Servicers are detailed with Servicer ID, Name, Contact Information, and Location. Each repair is assigned to one servicer.

Employees are integral to MHHC’s operations, particularly Sales Specialists and Claims Specialists. General employee data includes Employee ID, Name, Position, Department, Hire Date, Salary, Phone Number, and Email. Sales Specialists require tracking of Sales Count, Average Sales Volume, and Profit Margin. Claims Specialists manage Assigned Claims, Approved Claims, and generate Claims Approval Reports. Employees can also run Reports, identified by Report ID, Report Date, Term (From-To), Report Type, and Owner.

Contracts define the conditions for policies. Each contract is tracked by Contract ID, Customer ID, Agent ID, Policy ID, Start Date, and End Date. Policies are categorized by Policy Category ID, Name, Description, and Status, and are further detailed with Plan ID, Product Name, Terms, and Wholesale Cost. Underwriters, identified by Underwriter ID, Name, Contact Information, and Covered States, agree upon multiple policies.

Invoices are tracked with Invoice ID, Amount, Date, Due Date, Payment Status, Payment Method, Discount, Type, Billing Address, and Notes. These records provide transparency and facilitate financial reporting.

Dealers are tracked with Dealer ID, Name, Phone, Email, and Address. Dealer Catalogs link dealers with policies and include Dealer Catalog ID, Policy ID, and Retail Price. Return Details are managed with Return ID, Date, Reason, and Description.

Customers provide feedback, which is tracked with Feedback ID, Rating, Notes, Feedback Date, Response Date, and Status. This data informs improvements in service quality.

The existing system faces issues like fragmented data storage, inefficiency in claims processing, inconsistent reporting, and limited scalability. A centralized database resolves these challenges by integrating data management, automating workflows, improving data accuracy, enabling advanced reporting, and supporting scalability. This ensures streamlined operations, better service quality, and sustained success for MHHC Enterprises.

# Chapter 2: Conceptual Schema

## 2A. ER DIAGRAM:

Visio File: [MHHC\_531\_ER\_diagram\_final\_report.vsdx](https://emailarizona-my.sharepoint.com/:u:/g/personal/arnavsingh_arizona_edu/EcVyo_aVSXxPp2uR2H8kqa4BLi43GyHD9sSRKzFx99n13A?e=AT6UHw)

A diagram of a flowchart

Description automatically generated

## 2B. Data Dictionary (Conceptual/ER)

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No | Schema Construct | Construct Meaning | Structure and or Constraints |
| 1 | **ASSET**  **AssetID**  **PlanID**  **Brand**  **Model**  **Serial#**  **Condition**  **MFG Labor Ends**  **MFG Part Ends**  **Retail Value** | *Entity Class, models Asset data*  *AssetID is an identifying attribute*  PlanID is a foreign key referencing Policy table.  Brand name of the asset (Eg. Apple)  Model of the asset (Eg: Iphone 10)  The serial number of the asset  The condition of the asset  The end date of liability for labor expenses for servicer  The end date of liability for part expenses for servicer  The retail value of the asset | Format: NOT NULL, UNIQUE  Format: NOT NULL,  Data type: String  Nullable  Data type: String  Nullable  Data type: Number  UNIQUE  Allowed values: {New, Used, Refurbished}, NOT NULL  Format: mm-dd-yyyy, NOT NULL  Format: mm-dd-yyyy, NOT NULL  Data type: Number |
| 2 | **CLAIM**  **ClaimID**  **AssetID**  **CustomerID**  **InvoiceID**  **Loss Date**  **Problem Description**  **Claim Amount** | *Entity Class, models Claim data*  *ClaimID is an identifying attribute*  AssetID is a foreign key referencing asset table.  CustomerID is a foreign key referencing customer table.  InvoiceID is a foreign key referencing invoice table.  The date that the issue occurred  The description of problem that occurred with the device  The amount the individual is asking to be paid | Format: 7Digits, NOT NULL, UNIQUE  Format: 7Digits, NOT NULL, UNIQUE  Format: 7Digits, NOT NULL, UNIQUE  Format: 7Digits, NOT NULL, UNIQUE  Format: mm-dd-yyyy, NOT NULL  Format: String, NOT NULL  Format: Data Type: Decimal (10,2)  Range: > 0. Not NULL |
| 3 | **CLAIM APPROVAL**  **ClaimID**  **EmployeeID**  **ApprovalStatus**  **ApprovalDate**  **ApprovalAmount**  **ReviewDate**  **Lastupdated** | *Weak Entity Class, models data from CLAIM and CLAIM SPECIALIST*  CLaimID and EmployeeID are the identifiers inherited from the parent attributes.  The status of the claim approval  The date the status was accepted and approved  Approved Amount in dollars  The date the status was most recently reviewed  The date that the approval status was last updated | Data Type: Number, NOT NULL< UNIQUE  Allowed Values: {Accepted, Pending, Denied}, NOT NULL  Format: mm-dd-yyyy  Data Type: Decimal (10,2)  Range: > 0. Not NULL  Format: mm-dd-yyyy, NOT NULL  Format: mm-dd-yyyy |
| 4 | **CLAIM TEAM**  **TeamID**  **ResellerID**  **ServicerID**  **DealerID**  **EmployeeID**  **ProcessTime**  **ProcessNotes**  **TeamSuccessRate**  **FormationDate**  **TeamStatus** | *Aggregate Class, models Claims Team data*  *Formed by SERVICER, CLAIM SPECIALIST, RESELLER, DEALER*  *TeamID is an identifying attribute*  ResellerID,  ServicerID,  DealerID,  EmployeeID are foreign keys referring to Reseller, servicer, dealer and employee table  The amount of time it takes the team to fully process a claim  Any notes about the team's process  The percentage of time a claim is approved, and product is repaired  The date the team was formed  The status of the team | Not NULL, Format: 7 Digits, NOT NULL, UNIQUE  Not NULL, Format: 7 Digits, NOT NULL, UNIQUE  Not NULL, Format: 7 Digits, NOT NULL, UNIQUE  Not NULL, Format: 7 Digits, NOT NULL, UNIQUE  Not NULL, Format: 7 Digits, NOT NULL, UNIQUE  Data Type: Numeric (2 decimal places)  Data Type: String (optional)  Data Type: Numeric (2 decimal places, 0-100 range)  Not NULL, Format: mm-dd-yyyy,  Data Type: String  Allowed values: {Active, Inactive}, NOT NULL |
| 5 | **CONTRACT**  **ContractID**  **PlanID**  **CustomerID**  **StartDate**  **EndDate**  **Purchase Date** | *Entity Class, models Contract data*  *ContractID is an identifying attribute*  PlanID is a foreign key referencing to Policy table.  CustomerID is a foreign key referencing to Customer table.  Date the extended warranty is set to start  Date the extended warranty is set to expire  Date the extended warratny was purchased | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  NOT NULL, Format: mm-dd-yyyy,  NOT NULL, Format: mm-dd-yyyy,  NOT NULL, Format: mm-dd-yyyy, |
| 6 | **CUSTOMER (Superclass)**  **CustomerID**  **Name (FirstName**  **, LastName)**  **Phone**  **Location (**  **Address, City, State, ZipCode)**  **Email**  **Type** | *Entity Class, models Customer data*  *CustomerID is an identifying attribute*  The name of the customer consists of first and last name  Stores phone numbers. Multiple phone numbers may be entered.  The primary address of the customer. It consists of address, city, state, zipcode, country of the customer  The primary email address of the customer  Type of customer | Not NULL, Format: 7 Digits, UNIQUE  Data Type: String, Not NULL  Format: 10 digits stored as a string,  Not NULL  Address: String: NOT NULL, City: Not NULL; State: 2-letter format, NOT NULL; Zip Code: 5-digit format, NOT NULL;  Not NULL, Format: Valid Email, Data Type: String  Data Type: String |
| A | **ENTERPRISE BUSINESS (Subclass1)**  **CustomerID**  **ResellerID**  **RepName**  **AccountManager** | *Sub Class, models Enterprise Business data*  *CustomerID is an identifier inherited from the superclass.*  *ResellerID is a foreign key referencing the Reseller.*  *Name of the representative*  *Name of the account manager* | Not NULL, Fromat: 7 Digits, UNIQUE  Not NULL, Fromat: 7 Digits, UNIQUE  Data type: String  Nullable  Data type: String  Nullable |
| B | **INDIVIDUAL CUSTOMER (Subclass2)**  **CustomerID**  **ResellerID**  **Occupation**  **Income**  **ReferralSource**  **Notes** | *Sub Class, models Individual Customer data*  *CustomerID is an identifying attribute.*  *ResellerID is a foreign key referencing the Reseller.*  Job of the individual customer  Income of the customer  Where the customer is referred from  Individual customer notes | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Data type: String  Data type: Number  Data type: String  Data type: Varchar |
| 7 | **DEALER**  **DealerID**  **DealerName**  **Phone**  **Location**  **(Address, City, State, ZipCode, Country)**  **Email** | *Entity Class, models Dealer data*  *DealerID is an identifying attribute*  The name of the dealer consists of first and last name  Stores phone numbers. Multiple phone numbers may be entered.  The primary address of the dealer. It consists of address, city, state, zipcode, country of the customer  The primary email address of the customer | Not NULL, Format: 7 Digits, UNIQUE  Data Type: String, Not NULL  Format: 10 digits stored as a string,  Not NULL  City: Not NULL; State: 2-letter format, NOT NULL; Zip Code: 5-digit format, NOT NULL;  Not NULL, Format: Valid Email, Data Type: String |
| 8 | **DEALER CATALOG**  DealerID  P**lan**ID  **RetailPrice** | *Weak Entity Class, models Dealer Catalog and Policy/Warrant*  DealerID and PolicyID are the foreign keys referencing Dealer and Policy repectively.  RetailPrice is the price of the policy sold to the dealer | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL |
| 9 | **EMPLOYEE (Superclass)**  **EmployeeID**  **Name (FirstName, LastName)**  **Position**  **Department**  **HireDate**  **Salary**  **PhoneNumber**  **Email** | *Entity Class, models Employee data*  *EmployeeID is an identifying attribute*  First name of the employee  Last name of the employee  The position or role of the employee  The department the employee resides in  The date the employee was hired  The salary of the employee  The phone number of the employee  The email address of the employee | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  Data Type: String  Data Type: String  Not NULL, Data Type: String  Format: mm-dd-yyyy,  Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Phone number: 10 digits stored  Not NULL, Format: Valid Email, Data Type: String |
| A | **CLAIM SPECIALIST (Subclass1)**  **EmployeeID**  **Skills**  **Certifications** | *Sub Class, models Claim Specialist data*  EmployeeID is the identifier inherited from its superclass.  Skills of the claim specialist  All the certifications that the claims specialist possess | Data Type: Integer  Range: >= 0, Not NULL  Data Type: String  Data Type: String |
| B | **SALES SPECIALIST (Subclass2)**  **EmployeeID**  **CommissionRate**  **PerformanceRate**  **SalesRegion**  **PositionTitle**  **SalesVolume** | *Sub class, models*  EmployeeID is the identifier inherited from its superclass  Percentage of commission for the sales specialist.  The rate of performance of the specialist in percentage.  The region sales specialist is in.  Position of the sales specialist.  The number of sales completed by sales specialist | Data Type: Integer  Range: >= 0, Not NULL  Data Type: Varchar  Not NULL  Data Type: Varchar  Not NULL  Data Type: String  Not NULL  Data Type: String  Not NULL  Data Type: Numeric, Not NULL |
| 10 | **ENTERPRISE CUSTOMER**  **ECID**  **CustomerID**  **Name (IndividualFName, IndividualLName)**  **IndividualEmail** | *Weak Entity Class, models Enterprise Customer data*  Partial identifier  CustomerId is the foreign key referencing customer.  The first name of the individual customer  The last name of the individual customer  The primary email address of the customer | Dta Type: Number  Not Null  Not NULL, Format: 7 Digits, UNIQUE  Data Type: String  Not NULL  Data Type: String  Not NULL  Not NULL, Format: Valid Email, Data Type: String |
| 11 | **FEEDBACK**  **FeedbackID**  **Rating**  **FeedbackNotes**  **FeedbackDate**  **ResponseDate**  **Status** | *Entity Class, models Feedback Data*  *FeedbackID is an identifying attribute*  The rating 1-10 on the level of service  The notes given during feedback  The date the feedback was given  The date the feedback was responded to  The status of the feedback | Not NULL, Format: 7 Digits, UNIQUE  Data Type: Numeric  It could be from 1 to 10.  Data Type: String  Format: mm-dd-yyyy  Format: mm-dd-yyyy  Not NULL, Data Type: String  The status could be Active, Pending Inactive. |
| 12 | **INVOICE**  **InvoiceID**  **InvoiceDate**  **InvoiceAmount**  **DueDate**  **PaymentStatus**  **PaymentMethod**  **Discount**  **InvoiceType**  **BillingAddress (State, City, Zipcode, Country)**  **Notes** | *Entity Class, models Invoice data*  *InvoiceID is an identifying attribute*  The date the invoice was produced  The amount in $ the invoice was billed for  The date the invoice is due  The status of payment for the invoice (Paid, Partially Paid, Not Paid)  The method used to pay off the invoice  The percentage discount used on the invoice  The specific type of invoice (Claims, Sales, Repairs)  The address that the invoice was billed to  Any additional notes that are included on the invoice | Not NULL, Format: 7 Digits, UNIQUE  NOT NULL, Format: mm-dd-yyyy,  Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Format: mm-dd-yyyy  Not NULL, Data Type: String  It could be successful, in process or unsuccessful  Not NULL, Data Type: String  Data Type: Stored as Numeric with 2 decimal places. Example:10.00  Not NULL, Data Type: String  Data Type: String  City: Not NULL; State: 2-letter format, NOT NULL; Zip Code: 5-digit format, NOT NULL  Characters |
| 13 | **POLICY CATEGORY (TYPING CLASS)**  **CategoryID**  **CategoryName**  **Description**  **Status** | *Typing Class, models Policy Category data*  *CategoryID is an identifying attribute*  The name of a category  The description of the category  The status of the category (Active, Inactive) | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  Data Type: String  Not NULL, Data Type: String  It could be either Active or Inactive |
| 14 | **POLICY/WARRANTY (INSTANCE)**  **PlanID**  **CategoryID**  **UnderwriterID**  **CustomerID**  **ProductName**  **Product Description**  **Terms**  **Wholesale Price** | *Instance Class, models Policy data*  *PlanID is an identifying attribute*  CategoryID, UnderwriterID, CustomerID are foreign keys referencing to Policy Category, Underwriter and Customer tables respectively.  The name of the policy (Sku)  The description of the policy  The amount of time in months that the policy is valid for  The cost of the policy as agreed upon by the underwriter | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  Data Type: String  Not NULL, Data Type: String  Data Type: Stored as Numeric with 2 decimal places. Example:99.99 |
| 15 | **REPAIR**  **RepairID**  **ServicerID**  **RepairDate**  **ResolutionDescription**  **Claim Amount**  **Labor $**  **Labor Description**  **Travel $**  **Shipping $**  **Repair Part – Multivalue**  **RepairDescription** | *Entity Class, models Repair data*  *RepairID is an identifying attribute*  *ServicerID is a foreign key referencing Servicer table.*  The date the repair occurred  The resulting nature of the devicer (Fixed, New Product Ordered)  The amount in $ of the claim  The amount in $ of the cost assigned to labour expenses for the repair  The description of the repair | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: mm-dd-yyyy  Data Type: String  Not NULL, Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL, Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Data Type: String  Not NULL, Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL, Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL, Data Type: String  Not NULL, Data Type: String |
| 16 | **REPORT**  **ReportID**  **EmployeeID**  **ReportDate**  **TermFrom**  **TermTo**  **ReportType**  **ReportOwner** | *Entity Class, models Report data*  *Unique identifier for the Report*  *Employeeid is a foreign key referencing employee table.*  The date the report was ran  The earliest date the report contains data from  The latest date the report contains information to  The type of report  The name of the person that ran the report | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: mm-dd-yyyy  Not NULL, Format: mm-dd-yyyy  Not NULL, Format: mm-dd-yyyy  Data Type: String, Not NULL  Data Type: String, Not NULL |
| 17 | **RESELLER (Superclass)**  **ResellerID**  **EmployeeID**  **CompanyName**  **ContactPerson**  **ContactEmail**  **ContactPhone**  **Location (Address, City, State, Zipcode, Country)**  **DiscountRate**  **PartnerStatus**  **PaymentTerms**  **AccountCreationDate**  **ContractExpiryDate**  **Type** | *Entity Class, models Reseller data*  Unique identifier for the reseller  Employeeid is a foreign key referencing employee.  The name of the reseller company  The name of the primary point of contact  The email address of the primary point of contact  The address of the business  The percentage in discount that a reseller is allowed to use  The status of the reseller  The agreed upon terms of payment  The date the reseller was accepted  The date the reseller is no longer allowed to sell Extended service contracts  Type of reseller. | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  Not NULL, Data Type: String  Not NULL, Format: Valid Email, Data Type: String  Not NULL, Format: 10 digits, Data Type: String  Not NULL, Data Type: String  Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL, Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL, Data Type: String  It could be Active, Pending, or Inactive  Not NULL, Data Type: String  Not NULL, Format: mm-dd-yyyy, Data Type: String  Not NULL, Format: mm-dd-yyyy,  Data Type: String |
| A | **AGENT (Subclass1)**  **ResellerID**  **AgentStatus**  **AgentNotes**  **Role**  **JoinedDate** | *Sub Class Class, models Agent Data*  *ResellerID is an identifying attribute inherited from the superclass.*  *Status of the agent.*  *Notes generated by the agent.*  *Job role name of the agent.*  *The date agent joined the role.* | Format: 8 Digits, NOT NULL, UNIQUE  Data Type: String  Data Type: String  Data Type: String  Not NULL, Format: mm-dd-yyyy |
| B | **ENTERPRISE RESELLER (Subclass2)**  **ResellerID**  **AccountStatus**  **RegistrationDate** | *Sub Class, models Enterprise Reseller data*  *ResellerID is an identifying attribute inherited from the superclass.*  The status of the reseller regarding selling contracts (Active, Inactive, Pending Approval)  The date the reseller signed up to sell contracts | Format: 8 Digits, NOT NULL, UNIQUE  Not NULL, Data Type: String  The status could be Active, Pending Inactive.  NOT NULL, Format: mm-dd-yyyy |
| 18 | **RESELLER CATALOG**  **ResellerID**  **PlanID**  **RetailPrice**  **AdditionalCoverage** | *Weak Entity Class, models Reseller Catalog data*  ResellerId is the identfier.  Planid is a foreign key referencing Policy.  *Retail price is the price of the contract as sold to the dealer*  *AdditionalCoverage references any additional notes that might be related to a specific policy for the reseller* | Format: 8 Digits, NOT NULL, UNIQUE  Format: 8 Digits, NOT NULL, UNIQUE  Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Data Type: String |
| 19 | **RETURN DETAILS**  **ReturnID**  **DealerID**  **ResellerID**  **ReturnDetails**  **ReturnDate**  **ReturnReason** | *Weak Entity Class, models data from Dealer and Reseller*  *ReturnID is a partially identifying attribute*  *DealerID and ResellerID are foreign keys to Dealer and Reseller resopectively.*  A detailed description of the returned item, including specifics about the product or service returned  The date the return occurred.  The reason for the return | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Data Type: String  Not NULL, Format: mm-dd-yyyy, Data Type: String  Not NULL, Data Type: String |
| 20 | **SERVICER**  **ServicerID**  **Company**  **Address**  **City**  **State**  **Zipcode**  **Contact (Fax, Phone, Email** | *Entity Class, models Servicer data*  *ServicerID is an identifying attribute*  The name of the company providing the service.  The primary address of the service provider's business.  The primary contact person for the service provider.  The fax number for the service provider.  The primary phone number for the service provider.  The email address for the service provider. | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  Data Type: String  City: Not NULL; State: 2-letter format, NOT NULL; Zip Code: 5-digit format, NOT NULL  Not NULL, Data Type: String  Fax: 10 digits  Phone number: 10 digits  Email: Valid Email |
| 21 | **UNDERWRITER**  **UnderwriterID**  **UnderwriterName**  **City**  **State**  **Zip**  **Email**  **Phone**  **CoveredStates** | *Entity Class, models Underwriter data*  *UnderwriterID is an Identifying attribute*  *The name of the business underwriter*  *The city the underwriter resides in*  *The state the underwriter resides in*  *The zip code that the underwriter resides in*  *The primary email address for the underwriter*  *The primary phone number for the underwriter*  *The states covered by the underwriter* | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  City: Not NULL  State: 2-letter format, NOT NULL  Zip Code: 5-digit format, NOT NULL  Not NULL, Format: Valid Email, Data Type: String  Not NULL, Format: 10 digits, Data Type: String  State: 2-letter format, NOT NULL |

# Chapter 3: Relational Schema

## 3A. ER to Relational Normalized Tables:

1. **ASSET** (AssetID, Brand, Model, SerialNo, Condition, MFGLaborEnds, MFGPartEnds, RetailValue, PlanID)

Foreign Key PlanID References (POLICY)

1. **CLAIM** (ClaimID, AssetID, CustomerID, InvoiceID, LossDate, ProblemDescription, ClaimAmount)

Foreign Key CustomerID References (CUSTOMER )  
Foreign Key AssetID References (ASSET)

Foreign Key InvoiceID References (INVOICE)

1. **CLAIM\_TEAM** (TeamID, ResellerID,ServicerID, DealerID, EmployeeID, ProcessTime, ProcessNotes, TeamSuccessRate, FromationDate, TeamStatus)

Foreign Key ResellerID References (RESELLER)  
Foreign Key ServicerID References (SERVICER)

Foreign Key DealerID References (DEALER)

Foreign Key EmployeeID References (EMPLOYEE)

1. **CONTRACT** (ContractID, StartDate, EndDate, PurchaseDate, CustomerID, PlanID)

FOREIGN KEY CUSTOMER\_ID REFERENCES (CUSTOMER )  
FOREIGN KEY PLANID REFERENCES (POLICY)

1. **CUSTOMER** (CUSTOMER\_ID, ADDRESS, ZIPCODE, FIRSTNAME, LASTNAME, EMAIL, TYPE)  
    **i) CUSTOMER\_PHONES** (CUSTOMER\_ID, PHONE)   
    FOREIGN KEY CUSTOMER\_ID REFERENCES (CUSTOMER)

**ii) INDIVIDUAL\_CUSTOMER** (CUSTOMER\_ID, RESELLER\_ID, OCCUPATION, INCOME, REFERRALSOURCE, NOTES)   
 FOREIGN KEY CUSTOMER\_ID REFERENCES (CUSTOMER)   
 FOREIGN KEY RESELLER\_ID REFERENCES (RESELLER)

**A) INDIVIDUAL\_CUSTCONTRACTS** (CUSTOMER\_ID, CONTRACTS, RESELLER\_ID)  
FOREIGN KEY CUSTOMER\_ID REFERENCES CUSTOMER  
FOREIGN KEY RESELLER\_ID REFERENCES RESELLER

1. **ENTERPRISE\_BUSINESS** (CUSTOMER\_ID, RESELLER\_ID, REPNAME, ACCOUNTMANAGER)   
   FOREIGN KEY CUSTOMER\_ID REFERENCES (CUSTOMER)   
   FOREIGN KEY RESELLER\_ID REFERENCES (RESELLER)
2. **DEALER** (DealerID, DealerName, Address, City, State, ZipCode, Country, Email)
   1. **DEALER\_PHONE** (DealerID, Phone)

FOREIGN KEY DEALERID REFERENCES (DEALER)

1. **DISCOUNT** (InvoiceID, Discount)

Foreign Key InvoiceID References (INVOICE)

1. **EMPLOYEE** (EmployeeID, Department, Email, HireDate, Position, Salary, FirstName, LastName)
   1. **EMPLOYEE\_PHONE** (EmployeeID, Phone)

FOREIGN KEY EMPLOYEEID REFERENCES (EMPLOYEE)

* 1. **CLAIM SPECIALIST** (EmployeeID, Skills, Certifications)

FOREIGN KEY EMPLOYEEID REFERENCES (EMPLOYEE)

* 1. **SALES SPECIALIST** (EmployeeID, CommissionRate, PerformanceRating, SalesRegion, PositionTitle)

FOREIGN KEY EMPLOYEEID REFERENCES (EMPLOYEE)

1. **ENTERPRISE\_CUSTOMER** (ECID, CustomerID, IndividualFName, IndividualLName, IndividualEmail)

FOREIGN KEY CUSTOMERID REFERENCES (CUSTOMER)

1. **FEEDBACK** (FeedbackID, Rating, FeedbackNotes, FeedbackDate, ResponseDate, Status)
2. **INVOICE** (InvoiceID, InvoiceDate, InvoiceAmount, DueDate, PaymentStatus, PaymentMethod, InvoiceType, Address, Zipcode, Notes, ServicerID)
3. **POLICY** (PlanID, CategoryID, UnderwriterID, CustomerID, ProductName, ProductDescription, Terms, WholesalePrice)

FOREIGN KEY CATEGORYID REFERENCES (POLICY\_CATEGORY)

FOREIGN KEY UNDERWRITERID REFERENCES (UNDERWRITER)

FOREIGN KEY CUSTOMERID REFERENCES (CUSTOMER)

1. **POLICY\_CATEGORY** (CategoryID, CategoryName, Description, Status)
2. **REPAIR** (RepairID, RepairDate, ResolutionDescription, LaborDescription, RepairDescription, ClaimAmount, LaborAmount, TravelAmount, ShippingAmount, ServicerID)

FOREIGN KEY SERVICERID REFERENCES (SERVICER)

* 1. **REPAIR\_PART** (RepairID, RepairPart)

FOREIGN KEY REPAIRID REFERENCES (REPAIR)

1. **REPORT** (ReportID, ReportType, ReportDate, TermFrom, TermTo, ReportOwner, EmployeeID)

FOREIGN KEY EMPLOYEEID REFERENCES (EMPLOYEE)

1. **RESELLER** (ResellerID, CompanyName, SalesVolume, ContactPerson, ContactEmail, PartnerStatus, PaymentTerms, DiscountRate, Address, City, Zipcode, Country, PaymentTerms, AccountCreationDate, ContractExpiryDate, EmployeeID, Type)

FOREIGN KEY EMPLOYEEID REFERENCES (EMPLOYEE)

1. **RESELLER\_CONTACT** (ResellerID, ContactPhone)

FOREIGN KEY RESELLERID REFERENCES (RESELLER)

* 1. **AGENT** (ResellerID, AgentStatus, AgentNotes, Role, JoinedDate)

FOREIGN KEY RESELLERID REFERENCES (RESELLER)

* 1. **ENTERPRISE\_RESELLER** (ResellerID*,* AccountStatus, RegistrationDate)

FOREIGN KEY RESELLERID REFERENCES (RESELLER)

1. **SERVICER** (ServicerID, Company, Address, City, State, Zipcode, Fax, Email)
   1. **SERVICER\_PHONE** (ServicerID, Phone)

FOREIGN KEY SERVICERID REFERENCES (SERVICER)

1. **UNDERWRITER** (UnderwriterrID, UnderwriterName, Address, City, City, State, Email, CoveredStates)

**a. UNDERWRITER\_CONTACT** (UnderwriterID, Phone)

FOREIGN KEY UNDERWRITERIDREFERENCES (UNDERWRITER)

WEAK ENTITY TABLES:

1. **CLAIM\_APPROVAL** (ClaimID, EmployeeID, ApprovalStatus, ApprovalDate, ApprovalAmount, ReviewDate, LastUpdated)

FOREIGN KEY CLAIMID REFERENCES (CLAIM)

FOREIGN KEY EMPLOYEEID REFERENCES (EMPLOYEE)

1. **DEALER\_CATALOG** (DealerID, PolicyID, RetailPrice)

FOREIGN KEY DEALERID REFERENCES (DEALER)

FOREIGN KEY PLANID REFERENCES (POLICY)

1. **ENTERPRISE\_CUSTOMER** (CustomerID, Sequence Number, IndividualFName, IndividualLName, IndividualEmail)

FOREIGN KEY CUSTOMER\_ID REFERENCES (CUSTOMER)

1. **RESELLER\_CATALOG** (ResellerID, PlanID, RetailPrice, AddionalCoverage)

FOREIGN KEY RESELLERID REFERENCES (RESELLER)

FOREIGN KEY PLANID REFERENCES (POLICY)

1. **RETURN\_DETAILS** (ReturnID, DealerID, ResellerID, ReturnDetails, ReturnDate, ReturnReason)

FOREIGN KEY DEALERID REFERENCES (DEALER)

FOREIGN KEY RESELLERID REFERENCES (RESELLER)

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No | Schema Construct | Construct Meaning | Structure and or Constraints |
| 1 | **ASSET**  **AssetID**  **PlanID**  **Brand**  **Model**  **Serial#**  **Condition**  **MFG Labor Ends**  **MFG Part Ends**  **Retail Value** | *Entity Class, models Asset data*  *AssetID is an identifying attribute*  PlanID is a foreign key referencing Policy table.  Brand name of the asset (Eg. Apple)  Model of the asset (Eg: Iphone 10)  The serial number of the asset  The condition of the asset  The end date of liability for labor expenses for servicer  The end date of liability for part expenses for servicer  The retail value of the asset | Format: NOT NULL, UNIQUE  Format: NOT NULL,  Data type: String  Nullable  Data type: String  Nullable  Data type: Number  UNIQUE  Allowed values: {New, Used, Refurbished}, NOT NULL  Format: mm-dd-yyyy, NOT NULL  Format: mm-dd-yyyy, NOT NULL  Data type: Number |
| 2 | **CLAIM**  **ClaimID**  **AssetID**  **CustomerID**  **InvoiceID**  **Loss Date**  **Problem Description**  **Claim Amount** | *Entity Class, models Claim data*  *ClaimID is an identifying attribute*  AssetID is a foreign key referencing asset table.  CustomerID is a foreign key referencing customer table.  InvoiceID is a foreign key referencing invoice table.  The date that the issue occurred  The description of problem that occurred with the device  The amount the individual is asking to be paid | Format: 7Digits, NOT NULL, UNIQUE  Format: 7Digits, NOT NULL, UNIQUE  Format: 7Digits, NOT NULL, UNIQUE  Format: 7Digits, NOT NULL, UNIQUE  Format: mm-dd-yyyy, NOT NULL  Format: String, NOT NULL  Format: Data Type: Decimal (10,2)  Range: > 0. Not NULL |
| 3 | **CLAIM APPROVAL**  **ClaimID**  **EmployeeID**  **ApprovalStatus**  **ApprovalDate**  **ApprovalAmount**  **ReviewDate**  **Lastupdated** | *Weak Entity Class, models data from CLAIM and CLAIM SPECIALIST*  CLaimID and EmployeeID are the identifiers inherited from the parent attributes.  The status of the claim approval  The date the status was accepted and approved  Approved Amount in dollars  The date the status was most recently reviewed  The date that the approval status was last updated | Data Type: Number, NOT NULL< UNIQUE  Allowed Values: {Accepted, Pending, Denied}, NOT NULL  Format: mm-dd-yyyy  Data Type: Decimal (10,2)  Range: > 0. Not NULL  Format: mm-dd-yyyy, NOT NULL  Format: mm-dd-yyyy |
| 4 | **CLAIM TEAM**  **TeamID**  **ResellerID**  **ServicerID**  **DealerID**  **EmployeeID**  **ProcessTime**  **ProcessNotes**  **TeamSuccessRate**  **FormationDate**  **TeamStatus** | *Aggregate Class, models Claims Team data*  *Formed by SERVICER, CLAIM SPECIALIST, RESELLER, DEALER*  *TeamID is an identifying attribute*  The amount of time it takes the team to fully process a claim  Any notes about the team's process  The percentage of time a claim is approved, and product is repaired  The date the team was formed  The status of the team | Not NULL, Format: 7 Digits, NOT NULL, UNIQUE  Not NULL, Format: 7 Digits, NOT NULL, UNIQUE  Not NULL, Format: 7 Digits, NOT NULL, UNIQUE  Not NULL, Format: 7 Digits, NOT NULL, UNIQUE  Not NULL, Format: 7 Digits, NOT NULL, UNIQUE  Data Type: Numeric (2 decimal places)  Data Type: String (optional)  Data Type: Numeric (2 decimal places, 0-100 range)  Not NULL, Format: mm-dd-yyyy,  Data Type: String  Allowed values: {Active, Inactive}, NOT NULL |
| 5 | **CONTRACT**  **ContractID**  **PlanID**  **CustomerID**  **StartDate**  **EndDate**  **Purchase Date** | *Entity Class, models Contract data*  *ContractID is an identifying attribute*  PlanID is a foreign key referencing to Policy table.  CustomerID is a foreign key referencing to Customer table.  Date the extended warranty is set to start  Date the extended warranty is set to expire  Date the extended warranty was purchased | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  NOT NULL, Format: mm-dd-yyyy,  NOT NULL, Format: mm-dd-yyyy,  NOT NULL, Format: mm-dd-yyyy, |
| 6 | **CUSTOMER (Superclass)**  **CustomerID**  **FirstName**  **LastName**  **Address**  **City**  **State**  **ZipCode**  **Email**  **Type** | *Entity Class, models Customer data*  *CustomerID is an identifying attribute*  The name of the customer consists of first and last name  Stores phone numbers. Multiple phone numbers may be entered.  The primary address of the customer. It consists of address, city, state, zipcode, country of the customer  The primary email address of the customer  Type of customer | Not NULL, Format: 7 Digits, UNIQUE  Data Type: String, Not NULL  Format: 10 digits stored as a string,  Not NULL  Address: String: NOT NULL, City: Not NULL; State: 2-letter format, NOT NULL; Zip Code: 5-digit format, NOT NULL;  Not NULL, Format: Valid Email, Data Type: String  Data Type: String |
| i) | **CUSTOMER\_PHONES**  **CustomerID**  **Phone** | *CustomerID is an identifier inherited from the customer.*  *Phone numbers of customers.* | Not NULL, Format: 7 Digits, UNIQUE  Data type: Numeric, NOT NULL, 10-digit |
| A | **ENTERPRISE BUSINESS (Subclass1)**  **CustomerID**  **ResellerID**  **RepName**  **AccountManager** | *Sub Class, models Enterprise Business data*  *CustomerID is an identifier inherited from the superclass.*  *ResellerID is a foreign key referencing the Reseller.*  *Name of the representative*  *Name of the account manager* | Not NULL, Fromat: 7 Digits, UNIQUE  Not NULL, Fromat: 7 Digits, UNIQUE  Data type: String  Nullable  Data type: String  Nullable |
| B | **INDIVIDUAL CUSTOMER (Subclass2)**  **CustomerID**  **ResellerID**  **Occupation**  **Income**  **ReferralSource**  **Notes** | *Sub Class, models Individual Customer data*  *CustomerID is an identifying attribute.*  *ResellerID is a foreign key referencing the Reseller.*  Job of the individual customer  Income of the customer  Where the customer is referred from  Individual customer notes | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Data type: String  Data type: Number  Data type: String  Data type: Varchar |
| 7 | **DEALER**  **DealerID**  **DealerName**  **Phone**  **Address**  **City**  **State**  **ZipCode**  **Country**  **Email** | *Entity Class, models Dealer data*  *DealerID is an identifying attribute*  The name of the dealer consists of first and last name  Stores phone numbers. Multiple phone numbers may be entered.  The primary address of the dealer. It consists of address, city, state, zipcode, country of the customer  The primary email address of the customer | Not NULL, Format: 7 Digits, UNIQUE  Data Type: String, Not NULL  Format: 10 digits stored as a string,  Not NULL  City: Not NULL; State: 2-letter format, NOT NULL; Zip Code: 5-digit format, NOT NULL;  Not NULL, Format: Valid Email, Data Type: String |
| i) | **DEALER\_PHONE**  **DealerID**  **Phone** | DealerID is foreign key referencing Dealer. | Not NULL, Format: 7 Digits, UNIQUE  Data type: Numeric, NOT NULL, 10-digit |
| 8 | **DISCOUNT**  **InvoiceID**  **Discount** | InvoiceID is foreign key referencing the invoice table. | Not NULL, Format: 7 Digits, UNIQUE  Data type: varchar, NOT NULL |
| 8 | **DEALER CATALOG**  **DealerID**  P**lan**ID  **RetailPrice** | *Weak Entity Class, models Dealer Catalog and Policy/Warrant*  DealerID and PolicyID are the foreign keys referencing Dealer and Policy repectively.  RetailPrice is the price of the policy sold to the dealer | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL |
| 9 | **EMPLOYEE (Superclass)**  **EmployeeID**  **FirstName**  **LastName**  **Position**  **Department**  **HireDate**  **Salary**  **Email** | *Entity Class, models Employee data*  *EmployeeID is an identifying attribute*  First name of the employee  Last name of the employee  The position or role of the employee  The department the employee resides in  The date the employee was hired  The salary of the employee  The email address of the employee | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  Data Type: String  Data Type: String  Not NULL, Data Type: String  Format: mm-dd-yyyy,  Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL, Format: Valid Email, Data Type: String |
| i) | **EMPLOYEE\_PHONE**  **EmployeeID**  **Phone** | *EmployeeID is foreign key referencing employee tablee* | Not NULL, Format: 7 Digits, UNIQUE  Data type: Numeric, NOT NULL, 10-digit |
| A | **CLAIM SPECIALIST (Subclass1)**  **EmployeeID**  **Skills**  **Certifications** | *Sub Class, models Claim Specialist data*  EmployeeID is the identifier inherited from its superclass.  Skills of the claim specialist  All the certifications that the claims specialist possess | Data Type: Integer  Range: >= 0, Not NULL  Data Type: String  Data Type: String |
| B | **SALES SPECIALIST (Subclass2)**  **EmployeeID**  **CommissionRate**  **PerformanceRate**  **SalesRegion**  **PositionTitle** | *Sub class, models*  EmployeeID is the identifier inherited from its superclass  Percentage of commission for the sales specialist.  The rate of performance of the specialist in percentage.  The region sales specialist is in.  Position of the sales specialist. | Data Type: Integer  Range: >= 0, Not NULL  Data Type: Varchar  Not NULL  Data Type: Varchar  Not NULL  Data Type: String  Not NULL  Data Type: String  Not NULL |
| 10 | **ENTERPRISE CUSTOMER**  **ECID**  **CustomerID**  **IndividualFName IndividualLName**  **IndividualEmail** | *Weak Entity Class, models Enterprise Customer data*  Partial identifier  CustomerId is the foreign key referencing customer.  The first name of the individual customer  The last name of the individual customer  The primary email address of the customer | Dta Type: Number  Not Null  Not NULL, Format: 7 Digits, UNIQUE  Data Type: String  Not NULL  Data Type: String  Not NULL  Not NULL, Format: Valid Email, Data Type: String |
| 11 | **FEEDBACK**  **FeedbackID**  **Rating**  **FeedbackNotes**  **FeedbackDate**  **ResponseDate**  **Status** | *Entity Class, models Feedback Data*  *FeedbackID is an identifying attribute*  The rating 1-10 on the level of service  The notes given during feedback  The date the feedback was given  The date the feedback was responded to  The status of the feedback | Not NULL, Format: 7 Digits, UNIQUE  Data Type: Numeric  It could be from 1 to 10.  Data Type: String  Format: mm-dd-yyyy  Format: mm-dd-yyyy  Not NULL, Data Type: String  The status could be Active, Pending Inactive. |
| 12 | **INVOICE**  **InvoiceID**  **InvoiceDate**  **InvoiceAmount**  **DueDate**  **PaymentStatus**  **PaymentMethod**  **Discount**  **InvoiceType**  **BillingAddress (State, City, Zipcode, Country)**  **Notes** | *Entity Class, models Invoice data*  *InvoiceID is an identifying attribute*  The date the invoice was produced  The amount in $ the invoice was billed for  The date the invoice is due  The status of payment for the invoice (Paid, Partially Paid, Not Paid)  The method used to pay off the invoice  The percentage discount used on the invoice  The specific type of invoice (Claims, Sales, Repairs)  The address that the invoice was billed to  Any additional notes that are included on the invoice | Not NULL, Format: 7 Digits, UNIQUE  NOT NULL, Format: mm-dd-yyyy,  Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Format: mm-dd-yyyy  Not NULL, Data Type: String  It could be successful, in process or unsuccessful  Not NULL, Data Type: String  Data Type: Stored as Numeric with 2 decimal places. Example:10.00  Not NULL, Data Type: String  Data Type: String  City: Not NULL; State: 2-letter format, NOT NULL; Zip Code: 5-digit format, NOT NULL  Characters |
| 13 | **POLICY CATEGORY (TYPING CLASS)**  **CategoryID**  **CategoryName**  **Description**  **Status** | *Typing Class, models Policy Category data*  *CategoryID is an identifying attribute*  The name of a category  The description of the category  The status of the category (Active, Inactive) | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  Data Type: String  Not NULL, Data Type: String  It could be either Active or Inactive |
| 14 | **POLICY/WARRANTY (INSTANCE)**  **PlanID**  **CategoryID**  **UnderwriterID**  **CustomerID**  **ProductName**  **Product Description**  **Terms**  **Wholesale Price** | *IClass, models Policy data*  *PlanID is an identifying attribute*  CategoryID, UnderwriterID, CustomerID are foreign keys referencing to Policy Category, Underwriter and Customer tables respectively.  The name of the policy (Sku)  The description of the policy  The amount of time in months that the policy is valid for  The cost of the policy as agreed upon by the underwriter | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  Data Type: String  Not NULL, Data Type: String  Data Type: Stored as Numeric with 2 decimal places. Example:99.99 |
| 15 | **REPAIR**  **RepairID**  **ServicerID**  **RepairDate**  **ResolutionDescription**  **Claim Amount**  **Labor $**  **Labor Description**  **Travel $**  **Shipping $**  **Repair Part – Multivalue**  **RepairDescription** | *Entity Class, models Repair data*  *RepairID is an identifying attribute*  *ServicerID is a foreign key referencing Servicer table.*  The date the repair occurred  The resulting nature of the devicer (Fixed, New Product Ordered)  The amount in $ of the claim  The amount in $ of the cost assigned to labour expenses for the repair  The description of the repair | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: mm-dd-yyyy  Data Type: String  Not NULL, Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL, Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Data Type: String  Not NULL, Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL, Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL, Data Type: String  Not NULL, Data Type: String |
| i) | **REPAIR\_PART**  **RepairID**  **RepairPart** | *Foreign key referencing Repair table*  *Part needed to be repaired* | Not NULL, Format: 7 Digits, UNIQUE  Data type: String,  Not Null |
| 16 | **REPORT**  **ReportID**  **EmployeeID**  **ReportDate**  **TermFrom**  **TermTo**  **ReportType**  **ReportOwner** | *Entity Class, models Report data*  *Unique identifier for the Report*  *Employeeid is a foreign key referencing employee table.*  The date the report was ran  The earliest date the report contains data from  The latest date the report contains information to  The type of report  The name of the person that ran the report | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  NULL, Format: mm-dd-yyyy  Not NULL, Format: mm-dd-yyyy  Not NULL, Format: mm-dd-yyyy  Data Type: String, Not NULL  Data Type: String, Not NULL |
| 17 | **RESELLER (Superclass)**  **ResellerID**  **EmployeeID**  **CompanyName**  **ContactPerson**  **ContactEmail**  **ContactPhone**  **Location (Address, City, State, Zipcode, Country)**  **DiscountRate**  **PartnerStatus**  **PaymentTerms**  **AccountCreationDate**  **ContractExpiryDate**  **Type** | *Entity Class, models Reseller data*  Unique identifier for the reseller  Employeeid is a foreign key referencing employee.  The name of the reseller company  The name of the primary point of contact  The email address of the primary point of contact  The address of the business  The percentage in discount that a reseller is allowed to use  The status of the reseller  The agreed upon terms of payment  The date the reseller was accepted  The date the reseller is no longer allowed to sell Extended service contracts  Type of reseller. | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  Not NULL, Data Type: String  Not NULL, Format: Valid Email, Data Type: String  Not NULL, Format: 10 digits, Data Type: String  Not NULL, Data Type: String  Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL, Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Not NULL, Data Type: String  It could be Active, Pending, or Inactive  Not NULL, Data Type: String  Not NULL, Format: mm-dd-yyyy, Data Type: String  Not NULL, Format: mm-dd-yyyy,  Data Type: String |
| i) | **RESELLER\_CONTACT**  **ResellerID**  **ContactPhone** | *ResellerID is a foreign key referencing reseller*  *Contact number of reseller* | Format: 8 Digits, NOT NULL, UNIQUE  Format: 10 digit,  NOT NULL |
| A | **AGENT (Subclass1)**  **ResellerID**  **AgentStatus**  **AgentNotes**  **Role**  **JoinedDate** | *Sub Class Class, models Agent Data*  *ResellerID is an identifying attribute inherited from the superclass.*  *Status of the agent.*  *Notes generated by the agent.*  *Job role name of the agent.*  *The date agent joined the role.* | Format: 8 Digits, NOT NULL, UNIQUE  Data Type: String  Data Type: String  Data Type: String  Not NULL, Format: mm-dd-yyyy |
| B | **ENTERPRISE RESELLER (Subclass2)**  **ResellerID**  **AccountStatus**  **RegistrationDate** | *Sub Class, models Enterprise Reseller data*  *ResellerID is an identifying attribute inherited from the superclass.*  The status of the reseller regarding selling contracts (Active, Inactive, Pending Approval)  The date the reseller signed up to sell contracts | Format: 8 Digits, NOT NULL, UNIQUE  Not NULL, Data Type: String  The status could be Active, Pending Inactive.  NOT NULL, Format: mm-dd-yyyy |
| 18 | **RESELLER CATALOG**  **ResellerID**  **PlanID**  **RetailPrice**  **AdditionalCoverage** | *Weak Entity Class, models Reseller Catalog data*  ResellerId is the identfier.  Planid is a foreign key referencing Policy.  *Retail price is the price of the contract as sold to the dealer*  *AdditionalCoverage references any additional notes that might be related to a specific policy for the reseller* | Format: 8 Digits, NOT NULL, UNIQUE  Format: 8 Digits, NOT NULL, UNIQUE  Data Type: Stored as Numeric with 2 decimal places. Example:99.99  Data Type: String |
| 19 | **RETURN DETAILS**  **ReturnID**  **DealerID**  **ResellerID**  **ReturnDetails**  **ReturnDate**  **ReturnReason** | *Weak Entity Class, models data from Dealer and Reseller*  *ReturnID is a partially identifying attribute*  *DealerID and ResellerID are foreign keys to Dealer and Reseller resopectively.*  A detailed description of the returned item, including specifics about the product or service returned  The date the return occurred.  The reason for the return | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Format: 7 Digits, UNIQUE  Data Type: String  Not NULL, Format: mm-dd-yyyy, Data Type: String  Not NULL, Data Type: String |
| 20 | **SERVICER**  **ServicerID**  **Company**  **Address**  **City**  **State**  **Zipcode**  **Fax**  **Email** | *Entity Class, models Servicer data*  *ServicerID is an identifying attribute*  The name of the company providing the service.  The primary address of the service provider's business.  The primary contact person for the service provider.  The fax number for the service provider.  The primary phone number for the service provider.  The email address for the service provider. | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  Data Type: String  City: Not NULL; State: 2-letter format, NOT NULL; Zip Code: 5-digit format, NOT NULL  Not NULL, Data Type: String  Fax: 10 digits  Phone number: 10 digits  Email: Valid Email |
| 22 | **SERVICER\_PHONE**  **ServicerID**  **Phone** | ServicerID is foreign key referencing Servicer.  Phone number of servicer | Not NULL, Format: 7 Digits, UNIQUE  Data type: Numeric, NOT NULL, 10-digit |
| 21 | **UNDERWRITER**  **UnderwriterID**  **UnderwriterName**  **City**  **State**  **Zip**  **Email**  **CoveredStates** | *Entity Class, models Underwriter data*  *UnderwriterID is an Identifying attribute*  *The name of the business underwriter*  *The city the underwriter resides in*  *The state the underwriter resides in*  *The zip code that the underwriter resides in*  *The primary email address for the underwriter*  *The primary phone number for the underwriter*  *The states covered by the underwriter* | Not NULL, Format: 7 Digits, UNIQUE  Not NULL, Data Type: String  City: Not NULL  State: 2-letter format, NOT NULL  Zip Code: 5-digit format, NOT NULL  Not NULL, Format: Valid Email, Data Type: String  Not NULL, Format: 10 digits, Data Type: String  State: 2-letter format, NOT NULL |
| i) | **UNDERWRITER\_CONTACT UnderwriterID**  **Phone** | UnderwriterID is foreign key referencing Underwriter.  Phone number of underwriter | Not NULL, Format: 7 Digits, UNIQUE  Data type: Numeric, NOT NULL, 10-digit |
| RELATIONSHIPS | | | |
| 1 | **Provides** | Relationship that models which AGENT provides policies to INDIVIDUAL CUSTOMER  This relationship models the interaction between agents and individual customers regarding policy provision | Individual Customer --[0:M] --<>--[1:1] Agent  It indicates that an individual customer buys policies from one and only one agent, and conversely, a new agent may not have provided policies to any individual customers yet. Each individual agent can be associated with multiple customers. |
| 2 | **Transacts with** | Relationship that models which ENTERPRISE RESELLER sells to ENTERPRISE BUSINESS  This relationship models the interaction between enterprise resellers and enterprise businesses concerning sales transactions. | Enterprise Business --[0:M] --<>--[1:1] Enterprise Reseller  It indicates that, a new enterprise reseller may not have conducted any sales to enterprise businesses yet. Each enterprise business can transact with multiple enterprise resellers, but each sale is associated with exactly one enterprise reseller. |
| 3 | **Made on** | Relationship that models the Claims made on Assets. | Asset--[1:1] --<>--[0:M] Claim  It signifies that each asset can have many associated claims and may not have a claim yet. A claim can belong to one and only one asset. |
| 4 | **Covers** | Relationship that models the Assets covered by Policy. | Asset--[0:M] --<>--[1:1] Policy  Each asset must have at least one policy and not more than that. A newly introduced policy could not be covering an asset, but also a policy could cover many assets at a time. |
| 5 | **Files** | Relationship that models the CLAIMS made by CUSTOMERS | Customer--[1:1] --<>--[0:M] Claim  A customer may or may not file claim at a time and if they do, there can multiple claims per customer. And a claim can be filed by one and only one customer at a time. |
| 6 | **Settled in** | Relationship that models the Settlements through INVOICE applied to CLAIMS  This relationship models how claims are settled through invoices | Invoice--[1:1] --<>--[0:1] Claim  An invoice must be settled in at least one claim. But a claim will have one and only one invoice. |
| 7 | **Reviews** | Relationship that models the CLAIMS that are reviewed by CLAIM SPECIALIST and status of approval is stored in CLAIMS APPROVAL | Claim Specialist--[1:1] --<>--[1:1] Claim  Weak Entity: CLAIM APPROVAL [0:1]  For each pair of Claim Specialist and Claim, there may be zero or one Claim Approval rows associated with them. This indicates that while each Claim is tied to a single Claim Specialist, it may or may not have an approval. |
| 8 | **Forms** | Aggregate Relationship that models the CLAIM TEAM which consists of SERVICER, DEALER, CLAIM SPECIALIST and RESELLER | This aggregate relationship models the composition of a claim team. A claim Team can have 1 or 0 dealers, only 1 servicer, only claim specialist, 0 or 1 reseller.  This allows for variability in the composition of each claim team. |
| 9 | **Conditions Defined By** | Relationship that models the CONTRACTS and POLICY/WARRANTY. The warranty conditions are specified within the contract, dictating how claims and responsibilities are handled. | Contract--[1:M] --<>--[1:1] Policy  Each contract will define one and only one policy  One policy can be defined in one or many contract. |
| 10 | **Supplies to** | Relationship that models the RESELLER supplies to DEALER  This relationship models the policy or warranty purchased by a dealer as well as any RETURN DETAILS | Reseller--[1:1] --<>--[1:1] Dealer  Weak Entity: Return Details - [0:M]  For each pair of Reseller and Dealer, there may be zero or one Return Details associated with them. The details when a product is returned is stored in the Return Details. This indicates that while each Reseller is linked to a single Dealer, it may or may not have associated Return Details. |
| 11 | **Buys** | Relationship that models the POLICY/WARRANTY bought by DEALER and their specific DEALER CATALOG | Policy/Warranty--[1:1] --<>--[1:1] Dealer  Weak Entity: Dealer Catalog - [0:1]  For each pair of Policy/Warranty and Dealer, there may be zero or one Dealer Catalog associated with them. This indicates that while each Policy/Warranty is linked to a single Dealer, it may or may not have an entry in the Dealer Catalog. |
| 12 | **Gets Assigned By** | This relationship outlines the assignment of contracts between the enterprise business and its enterprise customers. | EB--[1:1] --<>--[0:M] EC  Each enterprise business does not have to engage with an enterprise customer, but can have many, while each customer may engage with one and only one business. |
| 13 | **Offered by** | This relationship outlines the FEEDBACK offered by CUSTOMERS | Feedback--[0:M] --<>--[1:1] Customer  A customer can provide multiple feedback, but each feedback is linked to only one specific customer. Some customers may not have given any feedback yet. |
| 14 | **Generates** | This relationship outlines each INVOICE generated by the SERVICER | SERVICER --[1:1] --<>--[0:M] INVOICE  Indicates that each Servicer can create multiple Invoices for repairs but does not have to. Every Invoice is linked to one and only one servicer |
| 15 | **Acquires** | This relationship outlines the policy that Reseller acquires from MHHC, as well as their specific RESELLER CATALOG | RESELLER--[1:1] --<>--[1:1] POLICY  Weak Entity: Reseller Catalog - [0:1]  For each pair of Reseller and Policy, there may be zero or one Reseller Catalog associated with them. This indicates that while each Policy is linked to a single Reseller, it may or may not have an entry in the Reseller Catalog. |
| 16 | **Perform** | Relationship models REPAIRS performed by SERVICER | REPAIR --[0:M] --<>--[1:1] SERVICER  A Servicer can perform many Repairs but does not have to. Each Repair is done by a single Servicer. |
| 17 | **Undergo** | Relationship models ASSETS that undergo REPAIRS | REPAIR --[0:1] --<>--[1:1] ASSET  Each Asset can have at most one Repair, and each Repair is performed on a specific Asset. This is a one-to-one relationship, but optional for the Asset side (since an Asset may not always need repairs). |
| 18 | **Markets To** | Relationship models MHCC SALESPECIALIST marketing to RESELLER | SALESPECIALIST --[1:1] --<>--[0:M] RESELLER A Sales Specialist can market to many Resellers, but each Reseller can only be marketed to by one Sales Specialist. This is a one-to-many relationship where one Sales Specialist is linked to multiple Resellers. |
| 19 | **Agrees Upon** | Relationship models UNDERWRITER Agreeing upon set POLICIES | UNDERWRITER--[1:1] --<>--[0:M] POLICIES  An underwriter can agree upon many set policies but does not have to agree upon any. A Policy is agreed upon by one and only one underwriter. |
| 20 | **Runs** | Relationship models EMPLOYEES Running REPORTS | EMPLOYEES -- [1:1] --<>--[0:M] REPORTS  An employee can run many reports but does not have to run any. A report is run by one and only one employee. |

## 3B. DDL Appendix:

Table Creation:

//Reseller

CREATE TABLE RESELLER (

ResellerID NUMBER(10),

CompanyName VARCHAR2(100),

SalesVolume NUMBER(12,2),

ContactPerson VARCHAR2(100),

ContactEmail VARCHAR2(100),

PartnerStatus VARCHAR2(50),

PaymentStatus VARCHAR2(50),

DiscountRate NUMBER(5,2),

Address VARCHAR2(255),

City VARCHAR2(100),

ZipCode VARCHAR2(15),

State VARCHAR2(100),

Country VARCHAR2(100),

PaymentTerms VARCHAR2(255),

AccountCreationDate DATE,

ContractExpiryDate DATE,

Type VARCHAR2(50),

EmployeeID NUMBER(10),

DealerID NUMBER(10),

CONSTRAINT reseller\_pk PRIMARY KEY (ResellerID),

CONSTRAINT FK\_Reseller\_EmployeeID FOREIGN KEY (EmployeeID) REFERENCES EMPLOYEE (EmployeeID),

CONSTRAINT FK\_Reseller\_DealerID FOREIGN KEY (DealerID) REFERENCES DEALER(DealerID)

);

//RESELLER CONTACT

CREATE TABLE RESELLER\_CONTACT (

Reseller\_ID NUMBER(10),

ContactPhone VARCHAR2(15),

CONSTRAINT resellercontact\_pk PRIMARY KEY (Reseller\_ID, ContactPhone),

CONSTRAINT FK\_ResellerContact\_ResellerID FOREIGN KEY (Reseller\_ID) REFERENCES RESELLER (Reseller\_ID)

);

//DEALER

CREATE TABLE DEALER (

DealerID NUMBER10),

DealerName VARCHAR2(100),

Address VARCHAR2(100),

State VARCHAR2(100),

City VARCHAR2(100),

ZipCode VARCHAR2(15),

Country VARCHAR2(100),

Email VARCHAR2(100),

CONSTRAINT dealer\_pk PRIMARY KEY (DEALERID)

);

//DEALER\_PHONE

CREATE TABLE DEALER\_PHONE (

DealerID NUMBER(10),

Phone VARCHAR2(15),

CONSTRAINT dealerphone\_pk PRIMARY KEY (DEALERID, PHONE),

CONSTRAINT FK\_DealerPhone\_DealerID FOREIGN KEY (DealerID) REFERENCES DEALER (DealerID)

);

//INVOICE

CREATE TABLE INVOICE (

InvoiceID NUMBER(10),

InvoiceDate DATE,

InvoiceAmount NUMBER(12, 2),

DueDate DATE,

PaymentStatus VARCHAR2(50),

PaymentMethod VARCHAR2(50),

InvoiceType VARCHAR2(50),

Address VARCHAR2 (100),

State VARCHAR2(100),

    City VARCHAR2(100),

    ZipCode VARCHAR2(15),

    Country VARCHAR2(100)

Notes VARCHAR2(500),

Discount NUMBER (5,2),

CONSTRAINT invoice\_pk PRIMARY KEY (INVOICEID)

);

//UNDERWRITER

CREATE TABLE UNDERWRITER (

UnderwriteID NUMBER(10),

UnderwriterName VARCHAR2(100),

Address VARCHAR2(255),

City VARCHAR2(100),

ZipCode VARCHAR2(15),

State VARCHAR2(100),

Country VARCHAR2(100),

Email VARCHAR2(100),

CONSTRAINT underwriter\_pk PRIMARY KEY (UnderwriteID)

);

//UNDERWRITER CONTACT

CREATE TABLE UNDERWRITER\_CONTACT (

UnderwriteID NUMBER(10),

Phone VARCHAR2(15),

CONSTRAINT underwritercontact\_pk PRIMARY KEY (UnderwriteID),

CONSTRAINT FK\_UnderwriterContact\_UnderwriterID FOREIGN KEY (UnderwriteID) REFERENCES UNDERWRITER (UnderwriteID)

);

//POLICY CATEGORY

CREATE TABLE POLICY\_CATEGORY (

CategoryID NUMBER(10),

Status VARCHAR2(50),

Description VARCHAR2(255),

CategoryName VARCHAR2(100),

CONSTRAINT policycategory\_pk PRIMARY KEY (CategoryID)

);

//POLICY

CREATE TABLE POLICY (

PlanID NUMBER(10),

ProductDescription VARCHAR2(255),

WholesalePrice NUMBER(12,2),

Terms NUMBER(3),

ProductName VARCHAR2(100),

CategoryID NUMBER(10),

Customer\_ID NUMBER(10),

UnderwriterID NUMBER(10),

CONSTRAINT policy\_pk PRIMARY KEY (PlanID),

CONSTRAINT FK\_Policy\_CategoryID FOREIGN KEY (CategoryID) REFERENCES POLICY\_CATEGORY (CategoryID),

CONSTRAINT FK\_Policy\_UnderwriterID FOREIGN KEY (UnderwriterID) REFERENCES UNDERWRITER (UnderwriteID),

CONSTRAINT FK\_Policy\_CustomerID FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID);

);

//CATALOG\_POLICY

CREATE TABLE CATALOG\_POLICY (

PolicyID NUMBER(10),

RetailPrice NUMBER(12, 2),

CONSTRAINT catalogpolicy\_pk PRIMARY KEY (POLICYID)

);

//DEALER\_CATALOG

CREATE TABLE DEALER\_CATALOG (

DealerID NUMBER(10),

PlanID NUMBER(10),

RetailPrice NUMBER(12,2),

CONSTRAINT dealercatalog\_pk PRIMARY KEY (DealerID, PlanID),

CONSTRAINT FK\_DealerCatalog\_DealerID FOREIGN KEY (DealerID) REFERENCES DEALER (DealerID),

CONSTRAINT FK\_DealerCatalog\_PlanID FOREIGN KEY (PlanID) REFERENCES POLICY (PlanID)

);

//CUSTOMER

CREATE TABLE CUSTOMER (

CUSTOMER\_ID INT,

FIRST\_NAME VARCHAR(50) NOT NULL,

LAST\_NAME VARCHAR(50) NOT NULL,

Address VARCHAR2(250)

State VARCHAR2(100),

City VARCHAR2(100),

ZipCode VARCHAR2(15),

Country VARCHAR2(100),

EMAIL VARCHAR(100) NOT NULL UNIQUE,

RESELLER\_ID NUMBER(10)

CONSTRAINT customer\_pk PRIMARY KEY (CUSTOMER\_ID)

CONSTRAINT FK\_Customer\_RESELLERID FOREIGN KEY (RESELLER\_ID) REFERENCES CUSTOMER(RESELLER\_ID)

);

//CUSTOMER\_PHONES

CREATE TABLE CUSTOMER\_PHONES (

CUSTOMER\_ID INT,

PHONE VARCHAR(15) NOT NULL,

CONSTRAINT customerphones\_pk PRIMARY KEY (CUSTOMER\_ID, PHONE),

CONSTRAINT FK\_CustomerPhones\_CustomerD FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID)

);

//ENTERPRISE\_BUSINESS

CREATE TABLE ENTERPRISE\_BUSINESS (

CUSTOMER\_ID INT,

RESELLER\_ID INT,

BUSINESS\_NAME VARCHAR(100) NOT NULL,

SALES DECIMAL(15, 2) NOT NULL,

CONSTRAINT enterprisebusiness\_pk PRIMARY KEY (CustomerID),

CONSTRAINT FK\_EnterpriseBusiness\_CustomerID FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID),

CONSTRAINT FK\_ EnterpriseBusiness \_ResellerID FOREIGN KEY (RESELLER\_ID) REFERENCES RESELLER(RESELLER\_ID)

);

//ENTERPRISE CUSTOMER

CREATE TABLE ENTERPRISE\_CUSTOMER (

Customer\_ID NUMBER(10),

Sequence\_Number NUMBER(10),

IndividualFName VARCHAR2(50),

IndividualLName VARCHAR2(50),

IndividualEmail VARCHAR2(100),

CONSTRAINT enterprisecustomer\_pk PRIMARY KEY (CustomerID, Sequence\_Number),

CONSTRAINT FK\_EnterpriseCustomer\_CustomerID FOREIGN KEY (CustomerID) REFERENCES ENTERPRISE\_BUSINESS (CustomerID)

);

//ENTERPRISE RESELLER

CREATE TABLE ENTERPRISE\_RESELLER (

ResellerID NUMBER(10),

AccountStatus VARCHAR2(50),

RegistrationDate DATE,

Sales NUMBER(12,2),

CONSTRAINT enterprisereseller\_pk PRIMARY KEY (ResellerID),

CONSTRAINT FK\_EnterpriseReseller\_ResellerID FOREIGN KEY (ResellerID) REFERENCES RESELLER (ResellerID)

);

//INDIVIDUAL\_CUSTOMER

CREATE TABLE INDIVIDUAL\_CUSTOMER (

CUSTOMER\_ID INT,

RESELLER\_ID INT,

CLAIM\_ID INT NOT NULL,

TOTAL\_CLAIM DECIMAL(10,2) NOT NULL,

CONSTRAINT individualcustomer\_pk PRIMARY KEY (CUSTOMER\_ID),

CONSTRAINT FK\_IndividualCustomer\_CustomerID FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID),

CONSTRAINT FK\_IndividualCustomer\_ResellerID FOREIGN KEY (RESELLER\_ID) REFERENCES RESELLER(RESELLER\_ID)

);

//INDIVIDUAL\_CUSTCONTRACTS

CREATE TABLE INDIVIDUAL\_CUSTCONTRACTS (

CUSTOMER\_ID INT,

CONTRACTS INT NOT NULL,

CONSTRAINT individualcustcontracts\_pk PRIMARY KEY (CUSTOMER\_ID, CONTRACTS),

CONSTRAINT FK\_IndividualCustcontracts\_CustomerID FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID)

);

//EMPLOYEE

CREATE TABLE EMPLOYEE (

EmployeeID NUMBER(10),

Department VARCHAR2(100),

Email VARCHAR2(100),

HireDate DATE,

Position VARCHAR2(50),

Salary NUMBER(12,2),

FirstName VARCHAR2(50),

LastName VARCHAR2(50),

CONSTRAINT employee\_pk PRIMARY KEY (EmployeeID),

);

//EMPLOYEE PHONE

CREATE TABLE EMPLOYEE\_PHONE (

EmployeeID NUMBER(10),

Phone VARCHAR2(15),

CONSTRAINT employeephone\_pk PRIMARY KEY (EmployeeID,Phone),

CONSTRAINT FK\_EmployeePhone\_EmployeeID FOREIGN KEY (EmployeeID) REFERENCES EMPLOYEE (EmployeeID)

);

//CLAIM SPECIALIST

CREATE TABLE CLAIM\_SPECIALIST (

EmployeeID NUMBER(10),

AssignedClaims NUMBER(10),

ApprovedClaims NUMBER(10),

CONSTRAINT claimspecialist\_pk PRIMARY KEY (EmployeeID),

CONSTRAINT FK\_ClaimSpecialist\_EmployeeID FOREIGN KEY (EmployeeID) REFERENCES EMPLOYEE (EmployeeID)

);

//AGENT

CREATE TABLE AGENT (

RESELLERID NUMBER(10) NOT NULL,

AGENTSTATUS VARCHAR2(50),

AGENTNOTES VARCHAR2(255),

ROLE VARCHAR2(50),

JOINEDDATE DATE,

CONSTRAINT AGENT\_PK PRIMARY KEY (RESELLERID)

);

//CONTRACT

CREATE TABLE CONTRACT (

CONTRACT\_ID INT,

START\_DATE DATE NOT NULL,

END\_DATE DATE NOT NULL,

PLANID INT,

CONSTRAINT contract\_pk PRIMARY KEY (Contract\_ID),

CONSTRAINT FK\_Contract\_PolicyID FOREIGN KEY (PLANID) REFERENCES POLICY(PLANID)

);

//SERVICER

CREATE TABLE SERVICER (

ServicerID NUMBER(10),

Company VARCHAR2(100),

Address VARCHAR2(100),

State VARCHAR2(100),

City VARCHAR2(100),

ZipCode VARCHAR2(15),

Country VARCHAR2(100),

Fax VARCHAR2(15),

Email VARCHAR2(100),

CONSTRAINT servicer\_pk PRIMARY KEY (SERVICERID)

);

//SERVICER\_PHONE

CREATE TABLE SERVICER\_PHONE (

ServicerID NUMBER(10),

Phone VARCHAR2(15),

CONSTRAINT servicerphone\_pk PRIMARY KEY (SERVICERID, PHONE),

CONSTRAINT FK\_ServicerPhone\_ServicerID FOREIGN KEY (SERVICERID) REFERENCES SERVICER (ServicerID)

);

//CLAIM TEAM

CREATE TABLE CLAIM\_TEAM (

TEAM\_ID NUMBER(10),

FORMATION\_DATE DATE,

AVG\_PROCESS\_TIME NUMBER(5,2),

TOTAL\_CLAIM\_AMOUNT NUMBER(12,2),

TEAM\_STATUS VARCHAR2(50),

PROCESS\_NOTES VARCHAR2(500),

TEAM\_SUCCESS\_RATES NUMBER(5,2),

SERVICERID NUMBER(10),

DEALERID NUMBER(10),

RESELLER\_ID NUMBER(10),

SPECIALIST\_ID NUMBER(10),

CONSTRAINT claimteam\_pk PRIMARY KEY (TEAM\_ID),

CONSTRAINT FK\_ClaimTeam\_ServicerID FOREIGN KEY (SERVICERID) REFERENCES SERVICER (SERVICERID),

CONSTRAINT FK\_ClaimTeam\_DealerID FOREIGN KEY (DEALERID) REFERENCES DEALER (DEALERID),

CONSTRAINT FK\_ClaimTeam\_SpecialistID FOREIGN KEY (SPECIALIST\_ID) REFERENCES CLAIM\_SPECIALIST (EMPLOYEEID)

);

//ASSET

CREATE TABLE ASSET (

AssetID NUMBER(10),

Brand VARCHAR2(100),

Condition VARCHAR2(50),

MFG\_Labor\_Ends DATE,

MFG\_Part\_Ends DATE,

AssetModel VARCHAR2(100),

Retail\_Value NUMBER(12, 2),

Serial\_No VARCHAR2(50) UNIQUE,

PlanID NUMBER(10,0),

CONSTRAINT asset\_pk PRIMARY KEY (ASSETID),

CONSTRAINT FK\_Asset\_PlanID FOREIGN KEY (PlanID) REFERENCES Policy (PlanID)

);

//CLAIM

CREATE TABLE CLAIM (

CLAIM\_ID NUMBER(10),

PROBLEM\_DESCRIPTION VARCHAR2(500),

LOSS\_DATE DATE,

CUSTOMER\_ID NUMBER(10),

ASSETID NUMBER(10),

INVOICEID NUMBER(10),

CLAIM\_AMOUNT (12,2)

CONSTRAINT claim\_pk PRIMARY KEY (CLAIM\_ID),

CONSTRAINT FK\_Claim\_CustomerID FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER (CUSTOMER\_ID),

CONSTRAINT FK\_Claim\_AssetID FOREIGN KEY (ASSETID) REFERENCES ASSET (ASSETID),

CONSTRAINT FK\_Claim\_InvoiceID FOREIGN KEY (INVOICEID) REFERENCES INVOICE (INVOICEID)

);

//CLAIM APPROVAL

CREATE TABLE CLAIM\_APPROVAL (

APPROVAL\_ID NUMBER(10),

APPROVAL\_STATUS VARCHAR2(50),

APPROVAL\_DATE DATE,

APPROVAL\_AMOUNT NUMBER(12,2),

REVIEW\_DATE DATE,

CLAIM\_ID NUMBER(10),

EMPLOYEEID (10),

LASTUPDATED TIMESTAMP(6)

SPECIALIST\_ID NUMBER(10),

CONSTRAINT claimapproval\_pk PRIMARY KEY (APPROVAL\_ID),

CONSTRAINT FK\_ClaimApproval\_ClaimID FOREIGN KEY (CLAIM\_ID) REFERENCES CLAIM (CLAIM\_ID),

CONSTRAINT FK\_ClaimApproval\_SpecialistID FOREIGN KEY (SPECIALIST\_ID) REFERENCES CLAIM\_SPECIALIST (EMPLOYEEID)

);

//REPAIR

CREATE TABLE REPAIR (

RepairID NUMBER(10),

RepairDate DATE,

ResolutionDescription VARCHAR2(255),

LaborDescription VARCHAR2(255),

RepairDescription VARCHAR2(255),

ClaimAmount NUMBER(12,2),

LaborAmount NUMBER(12,2),

TravelAmount NUMBER(12,2),

ShippingAmount NUMBER(12,2),

ServicerID NUMBER(10),

CONSTRAINT repair\_pk PRIMARY KEY (RepairID),

CONSTRAINT FK\_Repair\_ServicerID FOREIGN KEY (ServicerID) REFERENCES SERVICER (ServicerID)

);

//REPAIR PART

CREATE TABLE REPAIR\_PART (

RepairID NUMBER(10),

RepairPart VARCHAR2(255),

Repairdate Date

RESOLUTIONDESCRIPTION VARCHAR (255)

LABORDESCRIPTION VARCHAR2(255 BYTE)

CLAIMAMOUNT NUMBER(12,2)

LABORAMOUNT NUMBER(12,2)

TRAVELAMOUNT NUMBER(12,2)

SHIPPINGAMOUNT NUMBER(12,2)

SERVICERID NUMBER(10,0)

ASSETID NUMBER(10,0)

CONSTRAINT repairpart\_pk PRIMARY KEY (ReturnID, RepairPart),

CONSTRAINT FK\_RepairPart\_RepairID FOREIGN KEY (RepairID) REFERENCES REPAIR (RepairID)

);

//RETURN DETAILS

CREATE TABLE RETURN\_DETAILS (

RETURN\_ID NUMBER(10),

DEALER\_ID NUMBER(10),

RESELLER\_ID NUMBER(10),

RETURN\_DETAILS VARCHAR2(255),

RETURN\_DATE DATE,

RETURN\_REASON VARCHAR2(255),

CONSTRAINT returndetails\_pk PRIMARY KEY (RETURN\_ID, DEALER\_ID, RESELLER\_ID),

CONSTRAINT FK\_ReturnDetails\_DealerID FOREIGN KEY (DEALER\_ID) REFERENCES DEALER (DEALERID),

CONSTRAINT FK\_ReturnDetails\_ResellerID FOREIGN KEY (RESELLER\_ID) REFERENCES RESELLER (RESELLER\_ID)

);

//FEEDBACK

CREATE TABLE FEEDBACK (

FeedbackID NUMBER(10),

Rating NUMBER(2),

FeedbackNotes VARCHAR2(500),

FeedbackDate DATE,

ResponseDate DATE,

Status VARCHAR2(50),

CONSTRAINT feedback\_pk PRIMARY KEY (FEEDBACKID));

//REPORT

CREATE TABLE REPORT (

ReportID NUMBER(10),

ReportType VARCHAR2(50),

ReportDate DATE,

TermFrom DATE,

TermTo DATE,

ReportOwner VARCHAR2(100),

EmployeeID NUMBER(10),

CONSTRAINT report\_pk PRIMARY KEY (ReportID),

CONSTRAINT FK\_Report\_EmployeeID FOREIGN KEY (EmployeeID) REFERENCES EMPLOYEE (EmployeeID));

//DISCOUNT

CREATE TABLE DISCOUNT (

InvoiceID NUMBER(10),

Discount NUMBER(12, 2),

CONSTRAINT discount\_pk PRIMARY KEY (INVOICEID, DISCOUNT),

CONSTRAINT FK\_Discount\_InvoiceID FOREIGN KEY (InvoiceID) REFERENCES INVOICE (InvoiceID));

//SALES SPECIALIST

CREATE TABLE SALES\_SPECIALIST (

EmployeeID NUMBER(10),

SalesVolume NUMBER(12,2),

ProfitMargin NUMBER(5,2),

CONSTRAINT salesspecialist\_pk PRIMARY KEY (EmployeeID),

CONSTRAINT FK\_SalesSpecialist\_EmployeeID FOREIGN KEY(EmployeeID) REFERENCES EMPLOYEE(EmployeeID));

# Chapter 4: Queries

## Sales Specialist Performance Report​

**Generate a performance report of sales specialists showing their total sales, average profit margin and number of deals closed.​**  
WITH SalesPerformance AS (​  
    SELECT ​  
        ss.EmployeeID,​  
        e.FirstName || ' ' || e.LastName AS EmployeeName,​  
        ss.salesvolume AS TotalSales,​  
        Round(((ss.ProfitMargin)/ss.salesvolume)\*100,2) AS AvgProfitMargin​  
    FROM SALES\_SPECIALIST ss​  
    JOIN EMPLOYEE e ON ss.EmployeeID = e.EmployeeID​  
),​  
DealsClosed AS(​  
SELECT count(ss.EmployeeID) as TotalDealsClosed, ss.EmployeeID ​  
FROM RESELLER r ​  
JOIN SALES\_SPECIALIST ss on r.EmployeeID = ss.EmployeeID ​  
GROUP BY ss.EmployeeID)​  
SELECT ​  
    sp.EmployeeID,​  
    EmployeeName,​  
    TotalSales,​  
    AvgProfitMargin,​  
    TotalDealsClosed,​  
    CASE ​  
        WHEN AvgProfitMargin < 20 THEN '⚠️ Low Profit Margin'​  
        ELSE '✅ Good Performance'​  
    END AS PerformanceStatus​  
FROM SalesPerformance sp left join DealsClosed DC on sp.EmployeeID = dc.EmployeeID​  
ORDER BY TotalSales DESC;​

A screenshot of a computer

Description automatically generated

## Customer Claim History​

**Show each customer's total claims with the average claim amount and the highest claim they have filed. This helps in tracking claim history for risk assessment.​**

SELECT ​  
    c.Customer\_ID,​  
  c.first\_name || ' ' || c.last\_name AS CustomerName,​  
    COUNT(cl.claim\_id) AS TotalClaims,​  
    NVL(AVG(cl.Claim\_Amount),0) AS AvgClaimAmount,​  
    NVL((SELECT MAX(Claim\_Amount) FROM CLAIM WHERE Customer\_ID = c.Customer\_ID),0) AS HighestClaimAmount​  
FROM CUSTOMER c​  
LEFT JOIN CLAIM cl ON c.Customer\_ID = cl.Customer\_ID​  
GROUP BY c.Customer\_ID, c.first\_name, c.last\_name​  
ORDER BY TotalClaims DESC;​

A screenshot of a computer

Description automatically generated

## Ranking Claim Specialists by Approval Amount and Claim Count

**TOTAL\_APPROVED\_AMOUNT**: The total dollar amount of claims approved by the employee.

**TOTAL\_APPROVED\_CLAIMS**: The total number of claims approved by the employee.

This query ranks warranty specialists based on their performance in approving claims. It calculates the total approved amount and the total number of approved claims for each employee, then assigns a rank based on these metrics. The goal is providing insights for performance evaluations and operational improvements.

WITH specialist\_performance AS (

SELECT ca.EMPLOYEEID,

SUM(CASE WHEN ca.APPROVAL\_STATUS = 'Approved' THEN ca.APPROVAL\_AMOUNT ELSE 0 END) AS TOTAL\_APPROVED\_AMOUNT,

COUNT(CASE WHEN ca.APPROVAL\_STATUS = 'Approved' THEN 1 END) AS TOTAL\_APPROVED\_CLAIMS

FROM CLAIM\_APPROVAL ca

WHERE ca.APPROVAL\_STATUS IN ('Approved')

GROUP BY ca.EMPLOYEEID ),

ranked\_specialists AS (

SELECT sp.EMPLOYEEID, sp.TOTAL\_APPROVED\_AMOUNT, sp.TOTAL\_APPROVED\_CLAIMS, RANK() OVER (ORDER BY sp.TOTAL\_APPROVED\_AMOUNT DESC, sp.TOTAL\_APPROVED\_CLAIMS DESC) AS PERFORMANCE\_RANK

FROM specialist\_performance sp

)

SELECT

EMPLOYEEID,

TOTAL\_APPROVED\_AMOUNT,

TOTAL\_APPROVED\_CLAIMS,

PERFORMANCE\_RANK

FROM ranked\_specialists;

A screenshot of a computer

Description automatically generated

## Claim approval trend analysis over the past 90 days

**This query Assess recent trends in claim approvals. Identify any shifts in approval patterns or amounts. Monitor the volume of claims processed in different time periods. Evaluate the consistency of approval decisions over time**

SELECT

CASE

WHEN TRUNC(ca.LASTUPDATED) > TRUNC(SYSDATE) - 30 THEN 'Last 30 Days'

WHEN TRUNC(ca.LASTUPDATED) > TRUNC(SYSDATE) - 60 THEN '31-60 Days Ago'

WHEN TRUNC(ca.LASTUPDATED) > TRUNC(SYSDATE) - 90 THEN '61-90 Days Ago'

END AS TIME\_PERIOD,

ca.APPROVAL\_STATUS,

COUNT(\*) AS CLAIM\_COUNT,

AVG(ca.APPROVAL\_AMOUNT) AS AVG\_APPROVAL\_AMOUNT

FROM

CLAIM\_APPROVAL ca

GROUP BY

CASE

WHEN TRUNC(ca.LASTUPDATED) > TRUNC(SYSDATE) - 30 THEN 'Last 30 Days'

WHEN TRUNC(ca.LASTUPDATED) > TRUNC(SYSDATE) - 60 THEN '31-60 Days Ago'

WHEN TRUNC(ca.LASTUPDATED) > TRUNC(SYSDATE) - 90 THEN '61-90 Days Ago'

END,

ca.APPROVAL\_STATUS

ORDER BY TIME\_PERIOD, ca.APPROVAL\_STATUS;

A screenshot of a computer

Description automatically generated

## Customer Claim Analysis by State with Ranking and Subtotals

**This query is useful for analyzing customer claim data at multiple levels: Customer-Level Analysis: Identifies top customers in each state based on their claim amounts. State-Level Analysis: Provides subtotals for all customers within a state. Overall Analysis: Includes a grand total row showing the total claim amount across all states.**

SELECT cu.state, c.customer\_id, SUM(c.claim\_amount) AS total\_claim\_amount,

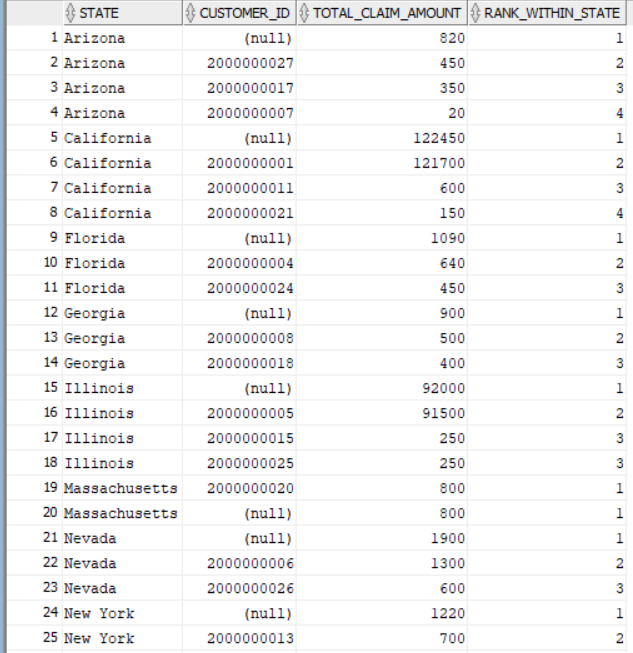
RANK() OVER (PARTITION BY cu.state ORDER BY SUM(c.claim\_amount) DESC) AS rank\_within\_state

FROM CLAIM c

JOIN CUSTOMER cu ON c.customer\_id = cu.customer\_id

GROUP BY ROLLUP(cu.state, c.customer\_id)

ORDER BY cu.state NULLS LAST, rank\_within\_state NULLS LAST, c.customer\_id;



## Query to Calculate Total Claim Amounts with ROLLUP for Hierarchical Summary by Reseller

**This query calculates total claim amounts by year and dealer with hierarchical subtotals and a grand total using ROLLUP. It joins the Claim, Customer, Reseller tables, groups data by the year of loss and reseller ID, and sums the claim\_amount. The ROLLUP generates subtotals for each year, totals for each dealer within a year, and an overall grand total.**

SELECT EXTRACT(YEAR FROM c.loss\_date),r.reseller\_id, sum(c.claim\_amount)

FROM Claim c JOIN Customer cu on c.customer\_id=cu.customer\_id

JOIN Reseller r on cu.reseller\_id = r.reseller\_id

Group by ROLLUP(EXTRACT(YEAR FROM c.loss\_date),r.reseller\_id);

A screenshot of a computer

Description automatically generated

## Find Customers with Above Average number of Claims

SELECT

c.CUSTOMER\_ID,

c.FIRST\_NAME || ' ' || c.LAST\_NAME AS CUSTOMER\_NAME,

COUNT(cl.CLAIM\_ID) AS CLAIM\_COUNT

FROM CUSTOMER c

JOIN CLAIM cl ON c.CUSTOMER\_ID = cl.CUSTOMER\_ID

GROUP BY c.CUSTOMER\_ID, c.FIRST\_NAME, c.LAST\_NAME

HAVING

COUNT(cl.CLAIM\_ID) > (

SELECT ceil(AVG(claim\_count))

FROM (

SELECT COUNT(CLAIM\_ID) AS claim\_count

FROM CLAIM

GROUP BY CUSTOMER\_ID

) avg\_claims

)

ORDER BY CLAIM\_COUNT DESC;

A screenshot of a computer

Description automatically generated

## Find Customers Who Have Never Subscribed to the Most Popular Plan

**List the names of customers who have never subscribed to the plan with the maximum number of contracts.**

SELECT CUSTOMER\_ID FROM CONTRACT WHERE CUSTOMER\_ID NOT IN (

SELECT DISTINCT C.CUSTOMER\_ID

FROM CONTRACT C

WHERE C.PLANID = (

SELECT PLANID

FROM (

SELECT PLANID, COUNT(\*) AS CONTRACT\_COUNT

FROM CONTRACT

GROUP BY PLANID

ORDER BY CONTRACT\_COUNT DESC

FETCH FIRST 1 ROWS ONLY)

)

);

A screenshot of a computer

Description automatically generated

## Long Processing Time Claims Analysis

This SQL query retrieves and analyzes insurance claims with processing times exceeding 30 days.

SELECT

ca.CLAIM\_ID,

c.CUSTOMER\_ID,

cu.FIRST\_NAME || ' ' || cu.LAST\_NAME AS CUSTOMER\_NAME,

c.LOSS\_DATE,

ca.APPROVAL\_DATE,

ca.APPROVAL\_DATE - c.LOSS\_DATE AS PROCESSING\_DAYS,

ca.APPROVAL\_STATUS,

ca.APPROVAL\_AMOUNT,

a.BRAND || ' ' || a.ASSETMODEL AS ASSET\_DETAILS

FROM CLAIM\_APPROVAL ca

JOIN CLAIM c ON ca.CLAIM\_ID = c.CLAIM\_ID

JOIN CUSTOMER cu ON c.CUSTOMER\_ID = cu.CUSTOMER\_ID

JOIN EMPLOYEE e ON ca.EMPLOYEEID = e.EMPLOYEEID

JOIN ASSET a ON c.ASSETID = a.ASSETID

WHERE ca.APPROVAL\_DATE - c.LOSS\_DATE > 30

ORDER BY PROCESSING\_DAYS DESC, ca.APPROVAL\_AMOUNT DESC;A screenshot of a computer

Description automatically generated

## Identify Top 3 Resellers by Sales in Each City

WITH ResellerSales AS (

SELECT City,

CompanyName,

SUM(SalesVolume) AS Total\_Sales,

RANK() OVER (PARTITION BY City ORDER BY SUM(SalesVolume) DESC) AS Sales\_Rank

FROM RESELLER

GROUP BY City, CompanyName

)

SELECT City,

CompanyName,

Total\_Sales,

Sales\_Rank

FROM ResellerSales

WHERE Sales\_Rank <= 3

ORDER BY City, Sales\_Rank;

A screenshot of a computer

Description automatically generated

# Chapter 5: Procedures And Triggers

### Procedure 1

set SERVEROUTPUT on;

CREATE OR REPLACE PROCEDURE manage\_claims\_status AS  
BEGIN  
    -- Condition 1: If the claim status is 'Needs more information' for over a month, send an alert and change status to 'Denied'  
    UPDATE claim\_approval  
    SET APPROVAL\_STATUS = 'Denied', LastUpdated = SYSDATE  
    WHERE APPROVAL\_STATUS = 'Needs more information'   
      AND LastUpdated < SYSDATE - INTERVAL '1' MONTH;

    -- Send an alert for the claims that were updated to 'Denied'  
    FOR rec IN (SELECT c.Claim\_ID, c.Customer\_ID FROM CLAIM c JOIN claim\_approval ca on c.claim\_id = ca.claim\_id  
                WHERE ca.Approval\_Status = 'Denied' AND LastUpdated = SYSDATE) LOOP  
        -- Example: send an alert or email (pseudo code, adjust for actual implementation)  
        -- CALL send\_alert('Ticket Dead', rec.ClaimID, rec.CustomerID);  
        DBMS\_OUTPUT.PUT\_LINE('Alert: ClaimID ' || rec.Claim\_ID || ' has been changed to Denied due to inactivity.');  
        -- Update in the audit table  
    END LOOP;  
  
    -- Condition 2: If the claim status is 'Pending' for over a month, assign it to a claim specialist for review  
    UPDATE claim\_approval  
    SET APPROVAL\_STATUS = 'Review', LastUpdated = SYSDATE  
    WHERE APPROVAL\_STATUS = 'Pending'  
      AND LastUpdated < SYSDATE - INTERVAL '1' MONTH;

    -- Send an alert for the claims that were assigned to a claim specialist  
    FOR rec IN (SELECT c.Claim\_ID, c.Customer\_ID FROM CLAIM c JOIN claim\_approval ca on c.claim\_id = ca.claim\_id  
                WHERE ca.APPROVAL\_STATUS = 'Review' AND LastUpdated = SYSDATE) LOOP  
        -- Example: notify a claim specialist (pseudo code, adjust for actual implementation)  
        -- CALL notify\_claim\_specialist('Claim Pending for Review', rec.ClaimID, rec.CustomerID);  
        DBMS\_OUTPUT.PUT\_LINE('Alert: ClaimID ' || rec.Claim\_ID || ' is now assigned to a claim specialist for review.');  
    END LOOP;

    -- Condition 3: Approved and Denied claims are set to 'Closed'  
    UPDATE claim\_approval  
    SET APPROVAL\_STATUS = 'Closed', LastUpdated = SYSDATE  
    WHERE APPROVAL\_STATUS IN ('Approved', 'Denied')  
      AND LastUpdated < SYSDATE - INTERVAL '1' MONTH;  
    COMMIT;  
  
    DBMS\_OUTPUT.PUT\_LINE('Claim statuses have been managed successfully.');  
END manage\_claims\_status;  
/

Before:  
A screenshot of a computer

Description automatically generated

After:

A screenshot of a computer

Description automatically generated

### Trigger 1 : Contract End Date Calculation

The trigger ensures that every contract inserted into the CONTRACT table has its END\_DATE automatically calculated based on the term length associated with the PLANID in the POLICY table. This eliminates the need for manual entry of the END\_DATE.

CREATE OR REPLACE TRIGGER BeforeInsertContract  
BEFORE INSERT ON CONTRACT  
FOR EACH ROW  
DECLARE   
    vTermLengthInYears NUMBER;  
    vTermLengthInMonths NUMBER;  
BEGIN  
    -- Get the term length in years from the POLICY table based on the PLANID  
    SELECT terms   
    INTO vTermLengthInYears   
    FROM POLICY   
    WHERE PLANID = :NEW.PLANID;

   -- Convert the term length from years to months  
    vTermLengthInMonths := vTermLengthInYears \* 12;

  -- Calculate the END\_DATE by adding the term length in months to the START\_DATE  
    :NEW.END\_DATE := ADD\_MONTHS(:NEW.START\_DATE, vTermLengthInMonths);  
  
END;  
/

Output:

A screenshot of a computer screen

Description automatically generated

### Trigger 2: Claim Validation Rules

The trigger TRG\_CLAIM\_VALIDATIONS is designed to perform multiple validations before inserting or updating a claim in the CLAIM table. It checks the following conditions and raises errors if any of them are violated:

* 1. Check if the Loss Date is within the contract's validity period: (Loss date is the date the issue happened)
  2. Check if the claim amount exceeds the asset cost

CREATE or REPLACE TRIGGER TRG\_CLAIM\_VALIDATIONS

BEFORE INSERT OR UPDATE ON CLAIM

FOR EACH ROW

DECLARE

v\_contract\_validity NUMBER; -- To check if loss date is within the contract start and end date

v\_asset\_cost NUMBER; -- To store the cost of the asset linked to the claim

BEGIN

-- Check if LossDate is outside the contract start and end date

SELECT COUNT(\*)

INTO v\_contract\_validity

FROM CLAIM c

JOIN CUSTOMER cu ON c.Customer\_ID = cu.Customer\_ID

JOIN CONTRACT co ON co.Customer\_ID = cu.Customer\_ID

JOIN POLICY p ON p.PlanID = co.PlanID

WHERE cu.Customer\_ID =:NEW.Customer\_ID

AND (:NEW.Loss\_Date < co.start\_date OR :NEW.Loss\_Date > co.end\_date);

IF v\_contract\_validity > 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Loss date outside contract validity period.');

END IF;

-- ️ Check if Claim Amount exceeds the Asset Cost

SELECT a.retail\_value

INTO v\_asset\_cost

FROM ASSET a

JOIN CLAIM c ON c.AssetID = a.AssetID

WHERE c.Claim\_ID = :NEW.Claim\_ID;

IF :NEW.Claim\_Amount > v\_asset\_cost THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Claim amount exceeds the asset cost.');

END IF;

END;

Error because loss date is outside contract validity period which is 01-OCT-2033

A screenshot of a computer

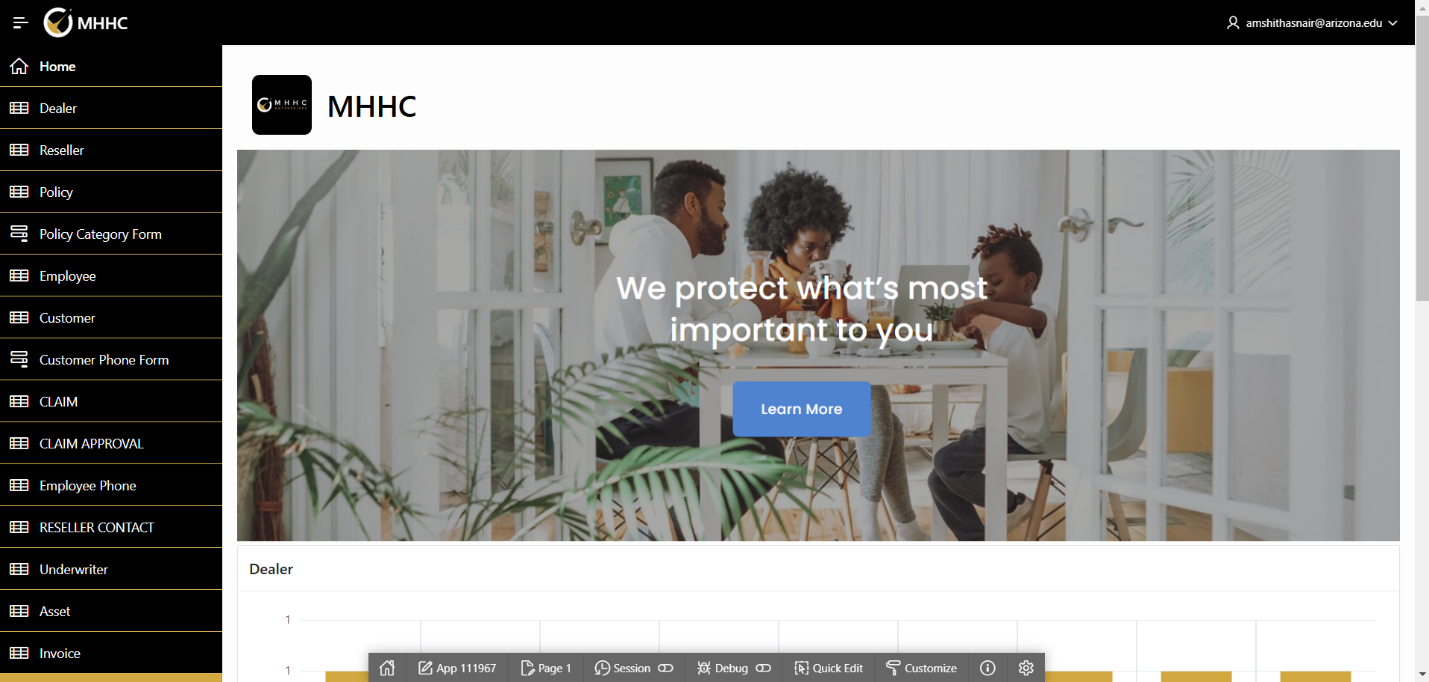
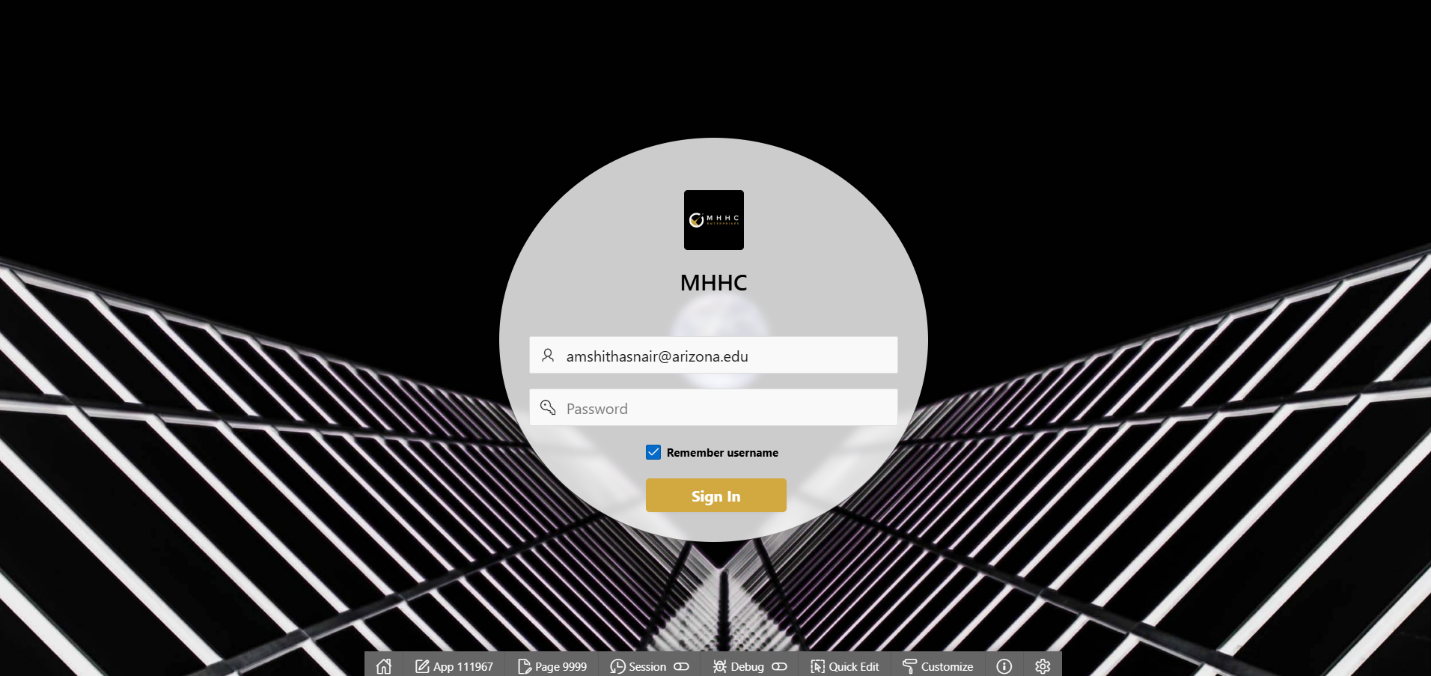
Description automatically generated

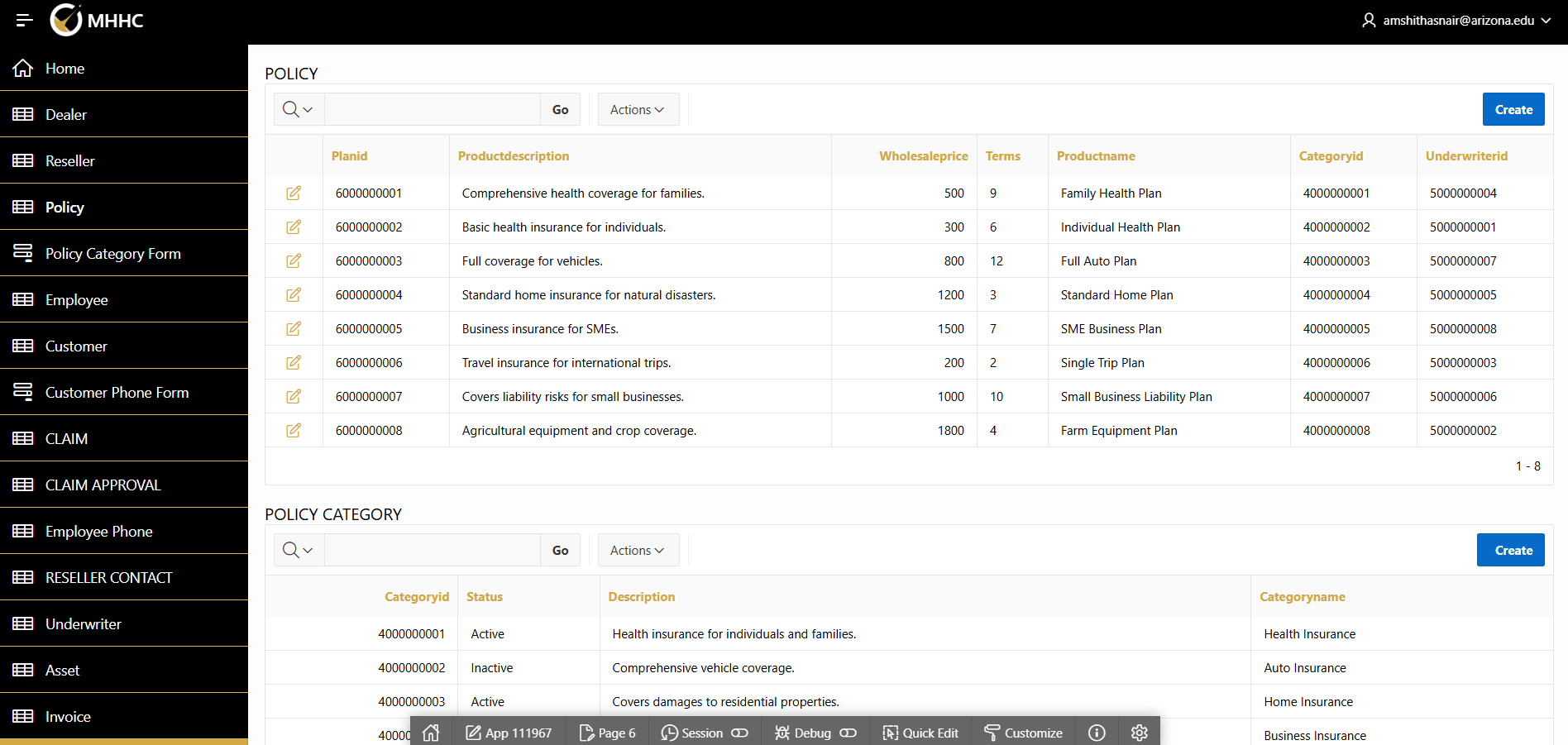
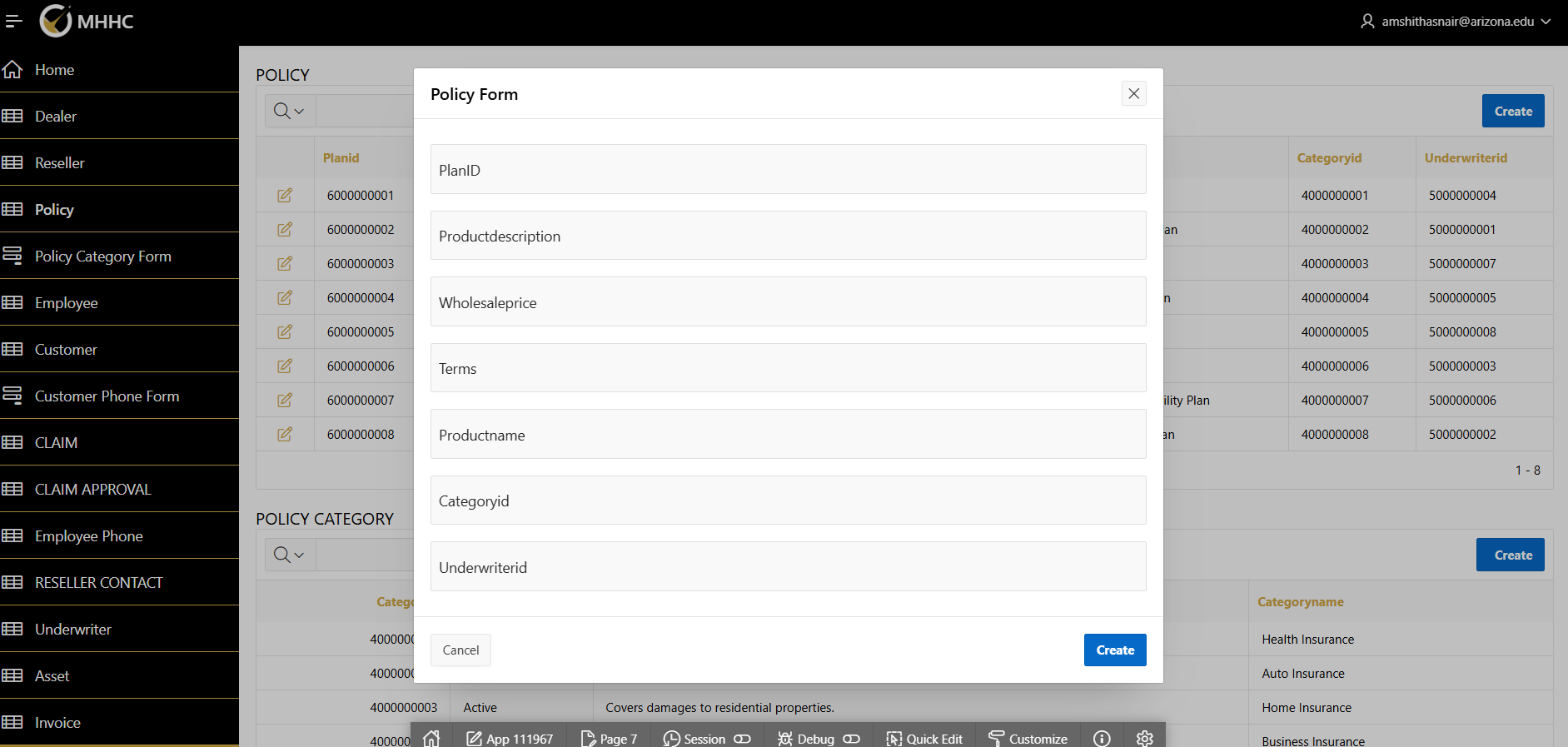
Error because Claim amount exceeds the asset cost. Asset cost is 1200

A screenshot of a computer

Description automatically generated

# Chapter 6: Frontend Overview



dry 

# Chapter 7: Implementation Plan

### Step 1: Preparation and Requirement Analysis

**Tasks:**

* Review the project design, database schema, and functional requirements from the report.
* Conduct meetings with stakeholders to finalize objectives and clarify ambiguities.
* Choose an appropriate database management platform, e.g., Oracle APEX for robust application development.
* Identify risks and prepare a mitigation plan for issues like schema mismatches or scalability challenges.

**Estimated Effort:**

* **Time:** 50 person-hours
* **Cost:** $4,000 (consultant fees and stakeholder meetings)

### Step 2: Environment Setup

**Tasks:**

* Provision cloud resources using Oracle Cloud Infrastructure (OCI) for Oracle APEX instances.
* Example: Oracle APEX Free Tier provides an environment with 1 OCPU and 20GB of storage at no cost. Paid instances with 2 OCPUs and 50GB storage cost approximately $360/month.
* Configure the database environment, including user roles, backup policies, and network settings.
* Install necessary tools like SQL Developer for managing Oracle databases.
* Implement monitoring tools available within OCI for database performance tracking.

**Estimated Effort:**

* **Time:** 40 person-hours
* **Cost:**
* **Cloud Costs:** $360/month for Oracle APEX (2 OCPUs and 50GB storage).
* **Personnel:** $2,500

### Step 3: Schema Creation

**Tasks:**

Execute SQL scripts to create tables, relationships, indexes, and constraints based on the finalized schema.

Validate the schema against test data for integrity and alignment.

Document schema for future reference and team onboarding.

**Estimated Effort:**

* **Time:** 35 person-hours
* **Cost:** $2,500

### Step 4: Data Migration

**Tasks:**

* Extract data from legacy systems or other sources.
* Transform the data to fit the new schema using Oracle Data Integrator (ODI) or SQL\*Loader.
* Load data into the database and validate data accuracy.
* Perform a pilot migration to a staging environment to identify and resolve issues.

**Estimated Effort:**

* **Time:** 60 person-hours
* **Cost:**
* **Personnel:** $3,500
* **ETL Tools:** $500 for Oracle Data Integrator usage costs.

### Step 5: Business Logic Implementation

**Tasks:**

* Develop stored procedures, triggers, and views for business logic requirements.
* Write unit tests for all procedures and validate performance metrics.
* Implement user-defined functions for reusable logic.

**Estimated Effort:**

* **Time:** 50 person-hours
* **Cost:** $3,000

### Step 6: Testing and Optimization

**Tasks:**

* Conduct unit testing for individual database components.
* Perform integration testing to ensure all components work together seamlessly.
* Optimize database performance through indexing and query tuning.
* Validate compliance with regulatory requirements like GDPR or HIPAA.

**Estimated Effort:**

* **Time:** 70 person-hours
* **Cost:** $4,000

### Step 7: Deployment

**Tasks:**

Deploy the database on the Oracle Cloud production environment.

Configure replication and failover mechanisms for high availability.

Validate the deployment with real-world workloads.

Implement monitoring and logging using OCI tools for early issue detection.

**Estimated Effort:**

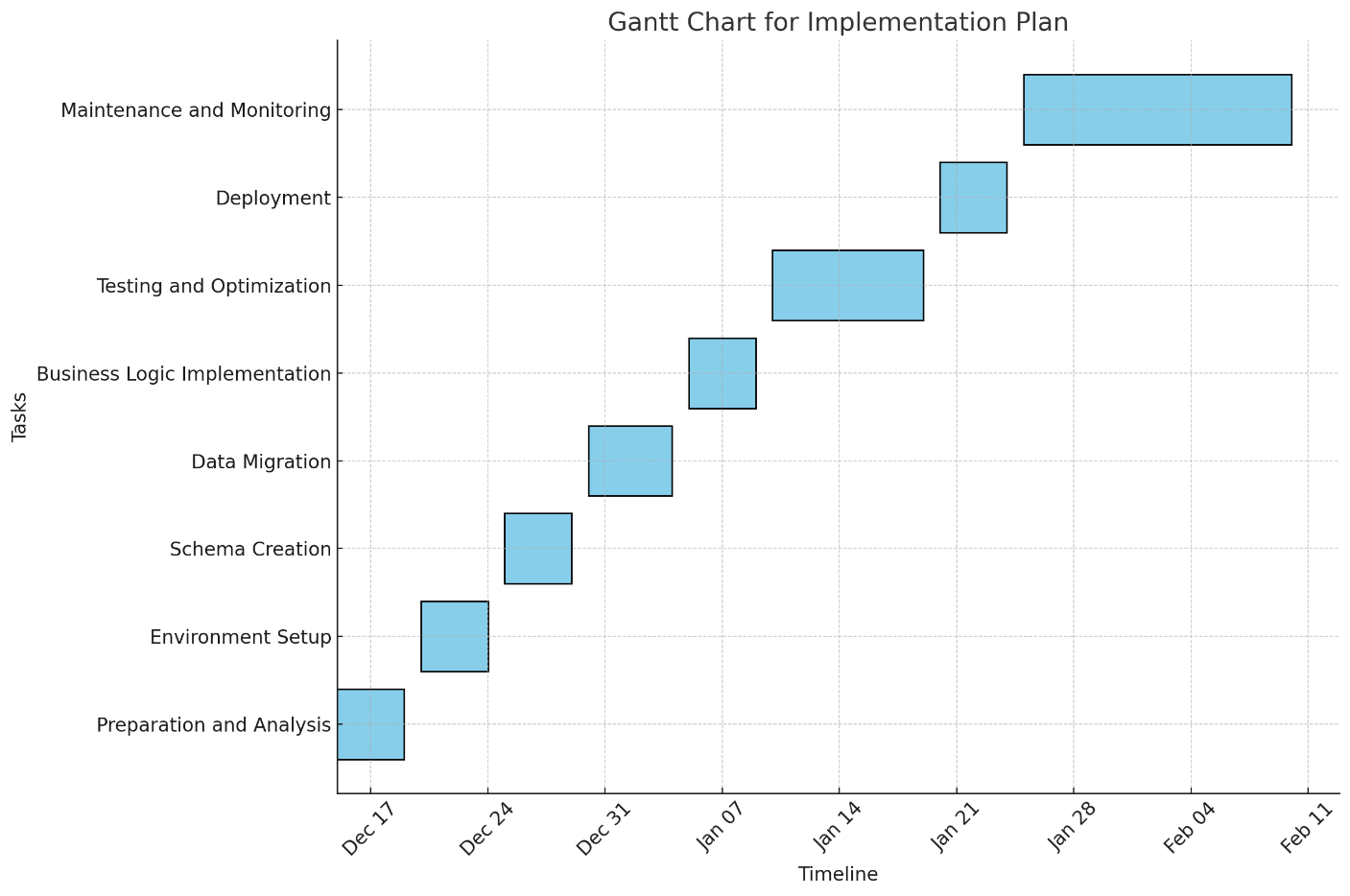
* **Time:** 40 person-hours
* **Cost:**
  + - **Personnel:** $2,500
    - **Infrastructure:** $100 for additional storage on Oracle Cloud Infrastructure.

### Step 8: Maintenance and Monitoring

**Tasks:**

* Continuously monitor database health and performance using Oracle APEX Monitor.
* Schedule automated backups and updates to keep the database secure.
* Allocate personnel for ongoing support and optimizations.
* **Estimated Effort:**
  + **Ongoing Cost:**
    - **Cloud Costs:** $360/month for Oracle APEX (paid tier).
    - **Personnel:** $1,500/month

Gantt Chart for Implementation Plan



### References

1. Oracle APEX Pricing: <https://apex.oracle.com/>
2. Oracle Cloud Infrastructure Pricing: <https://www.oracle.com/cloud/pricing/>

### Appendix A: Lessons Learned

**Lessons Learned from Project Implementation:**

1. **Effective Collaboration:**

* Close coordination between team members ensured seamless task transitions and minimized bottlenecks during critical phases.
* Regular meetings with stakeholders helped address ambiguities early and ensured alignment with project goals.

1. **Adaptability:**

* Encountering unforeseen challenges, such as schema mismatches, highlighted the importance of flexibility and quick problem-solving.
* Building modular SQL scripts allowed for faster adjustments to design changes.

1. **Importance of Testing:**

* Comprehensive testing, including integration and unit testing, revealed edge cases that would have caused significant issues post-deployment.
* Allocating extra time for testing saved potential downtime.
* Debugging queries was faster by using test cases.

1. **Cost and Resource Management:**

* Optimizing cloud resources by scaling based on actual workload reduced costs significantly during testing phases.
* Using Oracle APEX Free Tier for initial development minimized expenses before transitioning to the paid tier.

1. **Learnings from Other Teams:**

* Observing other teams' presentations emphasized the value of well-documented processes and visual aids to communicate complex ideas effectively.

1. **Documentation:**

* Creating detailed documentation for each step helped all the members adapt quickly to all phases of the project and ensured consistency in all milestones.