Team Error 404s

MIS 531: Enterprise Data Management

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

Contents

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.

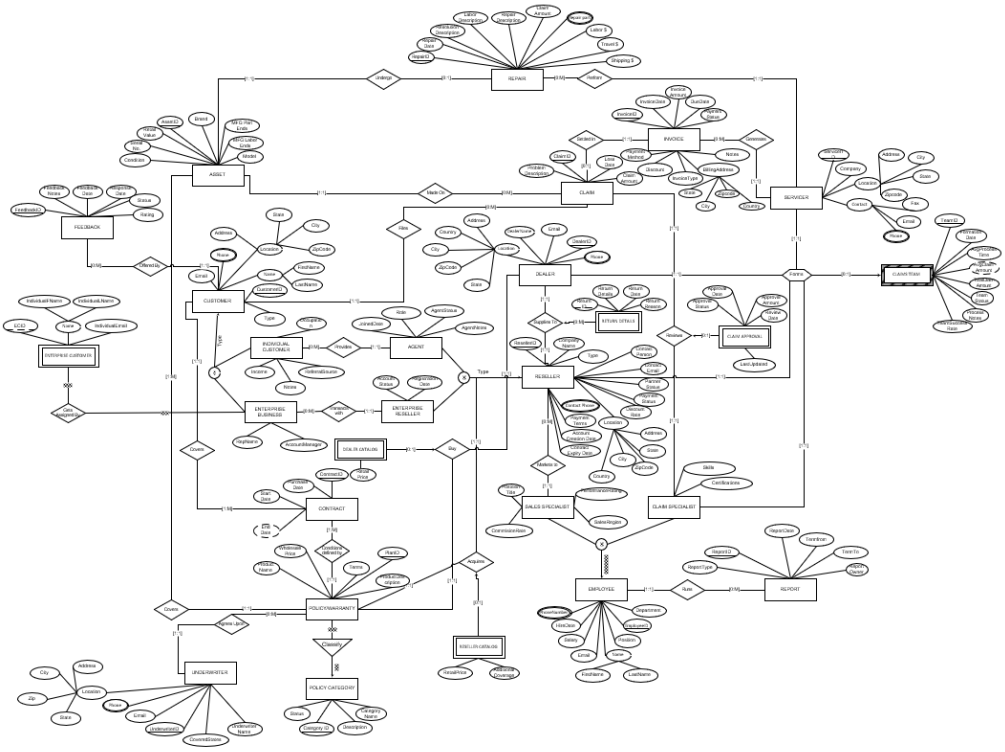
Customers provide feedback, which is tracked with Feedback ID, Rating, Notes, Feedback Date, Response Date, and Status. This data informs improvements in service quality.

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



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)	 Format: NOT NULL, UNIQUE Format: NOT NULL, Data type: String Nullable Data type: String Nullable Data type: Number UNIQUE Allowed values: {New, Used,

		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	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	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

		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	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**	*Entity Class, models Customer data*	Not NULL, Format: 7 Digits, UNIQUE

	Location (**Address, City, State, ZipCode)** **Email** **Type**	*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	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**	*Sub Class, models Individual Customer data*	Not NULL, Format: 7 Digits, UNIQUE

	Income **ReferralSource** **Notes**	*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 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.	Data Type: Integer Range: >= 0, Not NULL Data Type: String

		Skills of the claim specialist All the certifications that the claims specialist possess	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	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

		The primary email address of the customer	
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)	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

		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, 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	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

		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, 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**	*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	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

	ContractExpiryDate **Type**	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: 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.*	Format: 8 Digits, NOT NULL, UNIQUE Data Type: String Data Type: String Not NULL, Format: mm-dd-yyyy

		The date agent joined the role.	
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**	*Weak Entity Class, models data from Dealer and Reseller*	Not NULL, Format: 7 Digits, UNIQUE

	ReturnDetails **ReturnDate** **ReturnReason**	*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 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.	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

		The email address for the service provider.	
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)

2. **CLAIM** (ClaimID, AssetID, CustomerID, InvoiceID, LossDate, ProblemDescription, ClaimAmount)
   Foreign Key CustomerID References (CUSTOMER)
   Foreign Key AssetID References (ASSET)
   Foreign Key InvoiceID References (INVOICE)

3. **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)

4. **CONTRACT** (ContractID, StartDate, EndDate, PurchaseDate, CustomerID, PlanID)
   FOREIGN KEY CUSTOMER_ID REFERENCES (CUSTOMER)
   FOREIGN KEY PLANID REFERENCES (POLICY)

5. **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

i) **ENTERPRISE_BUSINESS** (CUSTOMER_ID, RESELLER_ID, REPNAME, ACCOUNTMANAGER)
FOREIGN KEY CUSTOMER_ID REFERENCES (CUSTOMER)
FOREIGN KEY RESELLER_ID REFERENCES (RESELLER)

6. **DEALER** (DealerID, DealerName, Address, City, State, ZipCode, Country, Email)
   a. **DEALER_PHONE** (DealerID, Phone)
      FOREIGN KEY DEALERID REFERENCES (DEALER)
7. **DISCOUNT** (InvoiceID, Discount)
   Foreign Key InvoiceID References (INVOICE)
8. **EMPLOYEE** (EmployeeID, Department, Email, HireDate, Position, Salary, FirstName, LastName)
   a. **EMPLOYEE_PHONE** (EmployeeID, Phone)
      FOREIGN KEY EMPLOYEEID REFERENCES (EMPLOYEE)
   b. **CLAIM SPECIALIST** (EmployeeID, Skills, Certifications)
      FOREIGN KEY EMPLOYEEID REFERENCES (EMPLOYEE)
   c. **SALES SPECIALIST** (EmployeeID, CommissionRate, PerformanceRating, SalesRegion, PositionTitle)
      FOREIGN KEY EMPLOYEEID REFERENCES (EMPLOYEE)
9. **ENTERPRISE_CUSTOMER** (ECID, CustomerID, IndividualFName, IndividualLName, IndividualEmail)
   FOREIGN KEY CUSTOMERID REFERENCES (CUSTOMER)
10. **FEEDBACK** (FeedbackID, Rating, FeedbackNotes, FeedbackDate, ResponseDate, Status)
11. **INVOICE** (InvoiceID, InvoiceDate, InvoiceAmount, DueDate, PaymentStatus, PaymentMethod, InvoiceType, Address, Zipcode, Notes, ServicerID)
12. **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)
13. **POLICY_CATEGORY** (CategoryID, CategoryName, Description, Status)
14. **REPAIR** (RepairID, RepairDate, ResolutionDescription, LaborDescription, RepairDescription, ClaimAmount, LaborAmount, TravelAmount, ShippingAmount, ServicerID)
    FOREIGN KEY SERVICERID REFERENCES (SERVICER)
    a. **REPAIR_PART** (RepairID, RepairPart)
       FOREIGN KEY REPAIRID REFERENCES (REPAIR)
15. **REPORT** (ReportID, ReportType, ReportDate, TermFrom, TermTo, ReportOwner, EmployeeID)
    FOREIGN KEY EMPLOYEEID REFERENCES (EMPLOYEE)

16. **RESELLER** (ResellerID, CompanyName, SalesVolume, ContactPerson, ContactEmail, PartnerStatus, PaymentTerms, DiscountRate, Address, City, Zipcode, Country, PaymentTerms, AccountCreationDate, ContractExpiryDate, EmployeeID, Type)
    FOREIGN KEY EMPLOYEEID REFERENCES (EMPLOYEE)
    i) **RESELLER_CONTACT** (ResellerID, ContactPhone)
        FOREIGN KEY RESELLERID REFERENCES (RESELLER)
    a. **AGENT** (ResellerID, AgentStatus, AgentNotes, Role, JoinedDate)
        FOREIGN KEY RESELLERID REFERENCES (RESELLER)
    b. **ENTERPRISE_RESELLER** (ResellerID, AccountStatus, RegistrationDate)
        FOREIGN KEY RESELLERID REFERENCES (RESELLER)
17. **SERVICER** (ServicerID, Company, Address, City, State, Zipcode, Fax, Email)
    a. **SERVICER_PHONE** (ServicerID, Phone)
        FOREIGN KEY SERVICERID REFERENCES (SERVICER)
18. **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)

2. **DEALER_CATALOG** (DealerID, PolicyID, RetailPrice)
   FOREIGN KEY DEALERID REFERENCES (DEALER)
   FOREIGN KEY PLANID REFERENCES (POLICY)

3. **ENTERPRISE_CUSTOMER** (CustomerID, Sequence Number, IndividualFName, IndividualLName, IndividualEmail)
   FOREIGN KEY CUSTOMER_ID REFERENCES (CUSTOMER)

4. **RESELLER_CATALOG** (ResellerID, PlanID, RetailPrice, AddionalCoverage)
   FOREIGN KEY RESELLERID REFERENCES (RESELLER)
   FOREIGN KEY PLANID REFERENCES (POLICY)

5. **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**	*Entity Class, models Claim data* *ClaimID is an identifying attribute*	Format: 7Digits, NOT NULL, UNIQUE Format: 7Digits, NOT NULL, UNIQUE

	Claim Amount	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: 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	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

		The date the status was most recently reviewed The date that the approval status was last updated	
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**	*Entity Class, models Contract data*	

	PlanID **CustomerID** **StartDate** **EndDate** **Purchase Date**	*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	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

		consists of address, city, state, zipcode, country of the customer The primary email address of the customer Type of customer	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.*	Not NULL, Format: 7 Digits, UNIQUE Not NULL, Format: 7 Digits, UNIQUE Data type: String Data type: Number Data type: String Data type: Varchar

		Job of the individual customer Income of the customer Where the customer is referred from Individual customer notes	
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	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

		The email address of the employee	
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**	*Weak Entity Class, models Enterprise Customer data*	Dta Type: Number Not Null

	IndividualFName **IndividualLName** **IndividualEmail**	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	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**	*Entity Class, models Invoice data* *InvoiceID is an identifying attribute* The date the invoice was produced	Not NULL, Format: 7 Digits, UNIQUE NOT NULL, Format: mm-dd-yyyy, Data Type: Stored as Numeric with 2

	BillingAddress (State, City, Zipcode, Country) **Notes**	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	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,	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

		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, 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	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

		contains information to The type of report The name of the person that ran the report	
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	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

		longer allowed to sell Extended service contracts Type of reseller.	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)	Format: 8 Digits, NOT NULL, UNIQUE Not NULL, Data Type: String The status could be Active, Pending Inactive. NOT NULL, Format: mm-dd-yyyy

		The date the reseller signed up to sell contracts	
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	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

		item, including specifics about the product or service returned The date the return occurred. The reason for the return	
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	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

		customers regarding policy provision	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	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.

		responsibilities are handled.	
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

1. 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;
```

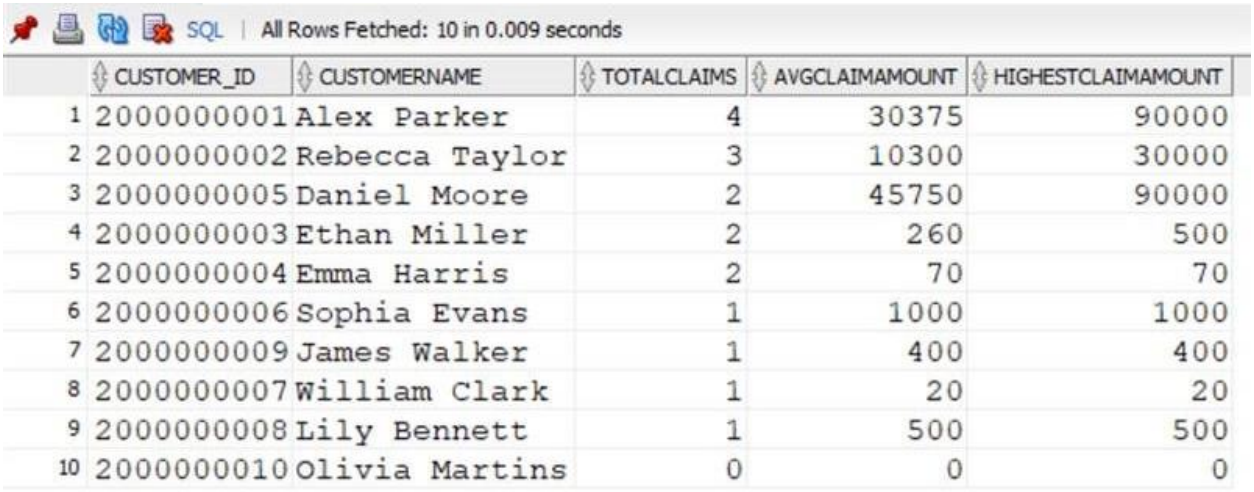
	EMPLOYEEID	EMPLOYEENAME	TOTALSALES	AVGPROFITMARGIN	TOTALDEALSCLOSED	PERFORMANCESTATUS
1	7000000010	Olivia Anderson	2500	32	8	✅ Good Performance
2	7000000001	Jonathan Doe	1500	33.33	5	✅ Good Performance
3	7000000003	Michael Johnson	1500	16.67	5	⚠ Low Profit Margin
4	7000000006	Sarah Brown	1200	19.17	4	⚠ Low Profit Margin
5	7000000005	Daniel Martinez	1000	20	3	✅ Good Performance

2. 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;
```



SQL | All Rows Fetched: 10 in 0.009 seconds

	CUSTOMER_ID	CUSTOMERNAME	TOTALCLAIMS	AVGCLAIMAMOUNT	HIGHESTCLAIMAMOUNT
1	2000000001	Alex Parker	4	30375	90000
2	2000000002	Rebecca Taylor	3	10300	30000
3	2000000005	Daniel Moore	2	45750	90000
4	2000000003	Ethan Miller	2	260	500
5	2000000004	Emma Harris	2	70	70
6	2000000006	Sophia Evans	1	1000	1000
7	2000000009	James Walker	1	400	400
8	2000000007	William Clark	1	20	20
9	2000000008	Lily Bennett	1	500	500
10	2000000010	Olivia Martins	0	0	0

3. 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;
```

	EMPLOYEEID	TOTAL_APPROVED_AMOUNT	TOTAL_APPROVED_CLAIMS	PERFORMANCE_RANK
1	7000000002	1100	2	1
2	7000000004	950	2	2
3	7000000009	800	1	3
4	7000000007	600	1	4

4. 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;
```

	TIME_PERIOD	APPROVAL_STATUS	CLAIM_COUNT	AVG_APPROVAL_AMOUNT
1	Last 30 Days	Approved	6	575
2	Last 30 Days	Closed	5	1200
3	Last 30 Days	Denied	5	0
4	Last 30 Days	Pending	3	(null)
5	Last 30 Days	Review	4	(null)

5. 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;
```

	STATE	CUSTOMER_ID	TOTAL_CLAIM_AMOUNT	RANK_WITHIN_STATE
1	Arizona	(null)	820	1
2	Arizona	2000000027	450	2
3	Arizona	2000000017	350	3
4	Arizona	2000000007	20	4
5	California	(null)	122450	1
6	California	2000000001	121700	2
7	California	2000000011	600	3
8	California	2000000021	150	4
9	Florida	(null)	1090	1
10	Florida	2000000004	640	2
11	Florida	2000000024	450	3
12	Georgia	(null)	900	1
13	Georgia	2000000008	500	2
14	Georgia	2000000018	400	3
15	Illinois	(null)	92000	1
16	Illinois	2000000005	91500	2
17	Illinois	2000000015	250	3
18	Illinois	2000000025	250	3
19	Massachusetts	2000000020	800	1
20	Massachusetts	(null)	800	1
21	Nevada	(null)	1900	1
22	Nevada	2000000006	1300	2
23	Nevada	2000000026	600	3
24	New York	(null)	1220	1
25	New York	2000000013	700	2

6. 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);
```

	EXTRACT(YEARFROMC.LOSS_DATE)	RESELLER_ID	SUM(C.CLAIM_AMOUNT)
1	(null)	1000000020	550
2	(null)	1000000026	600
3	(null)	1000000028	1500
4	(null)	1000000030	450
5	(null)	(null)	3100
6	2020	1000000016	150
7	2020	1000000019	450
8	2020	1000000021	200
9	2020	1000000028	90000
10	2020	1000000034	520
11	2020	1000000035	250
12	2020	(null)	91570
13	2021	1000000009	500
14	2021	1000000012	90000
15	2021	1000000017	20
16	2021	1000000020	400
17	2021	1000000032	700
18	2021	(null)	91620
19	2022	1000000009	70
20	2022	1000000012	200
21	2022	1000000020	150
22	2022	1000000021	500
23	2022	1000000024	500

7. 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;
```

	CUSTOMER_ID	CUSTOMER_NAME	CLAIM_COUNT
1	2000000001	Alex Parker	5
2	2000000002	Rebecca Taylor	4
3	2000000004	Emma Harris	3

8. 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)
      )
);
```

	CUSTOMER_ID
1	2000000002
2	2000000009
3	2000000005
4	2000000004
5	2000000010
6	2000000007
7	2000000001
8	2000000006

9. 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;
```

	CLAIM_ID	CUSTOMER_ID	CUSTOMER_NAME	LOSS_DATE	APPROVAL_DATE	PROCESSING_DAYS	APPROVAL_STATUS	APPROVAL_AMOUNT	ASSET_DETAILS
1	1100000001	2000000003	Ethan Miller	15-OCT-2020	01-OCT-2023	1081	Approved	800	Philips DreamStation CPAP
2	1100000012	2000000003	Ethan Miller	30-JUN-2020	15-JUN-2023	1080	Closed	3000	Honeywell Home Security System
3	1100000005	2000000005	Daniel Moore	20-DEC-2020	25-JUN-2023	917	Denied	0	Caterpillar Mini Excavator
4	1100000001	2000000003	Ethan Miller	15-OCT-2020	01-MAR-2023	867	Denied	0	Philips DreamStation CPAP
5	1100000014	2000000004	Emma Harris	18-MAR-2022	01-JUL-2024	836	Closed	700	Caterpillar Bulldozer
6	1100000007	2000000009	James Walker	14-MAY-2021	15-AUG-2023	823	Approved	450	Honeywell Home Security System
7	1100000004	2000000007	William Clark	13-SEP-2021	20-MAY-2023	614	Approved	600	Fitbit Charge 5
8	1100000013	2000000001	Alex Parker	05-DEC-2021	20-MAY-2023	531	Closed	1500	GE Small Business Refrigerator
9	1100000002	2000000008	Lily Bennett	28-NOV-2022	10-JUL-2023	224	Approved	700	GE Small Business Refrigerator
10	1100000015	2000000002	Rebecca Taylor	02-NOV-2023	01-JUN-2024	212	Closed	0	Kubota Tractor
11	1100000016	2000000001	Alex Parker	09-MAY-2024	01-NOV-2024	176	Closed	800	Whirlpool Refrigerator

10. 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;
```

	CITY	COMPANYNAME	TOTAL_SALES	SALES_RANK
21	Redwood City	Oracle	800000	1
22	Round Rock	Dell Technologies	500000	1
23	San Francisco	TechSoup	300000	1
24	San Francisco	Tech Solutions Inc.	150000	2
25	San Jose	Cisco	950000	1

Chapter 5: Procedures And Triggers

1. 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:

	APPROVAL_ID	APPROVAL_STATUS	APPROVAL_DATE	APPROVAL_AMOUNT	REVIEW_DATE	CLAIM_ID	EMPLOYEEID	LASTUPDATED
1	1400000030	Needs more information	(null)	(null)	01-JUL-2024	1100000010	7000000002	01-OCT-2024 12.00.00.000000000 AM
2	1400000031	Pending	(null)	(null)	01-SEP-2024	1100000011	7000000004	01-OCT-2024 12.00.00.000000000 AM
3	1400000032	Approved	15-JUN-2023	3000	(null)	1100000012	7000000007	01-OCT-2023 12.00.00.000000000 AM
4	1400000033	Denied	20-MAY-2023	1500	(null)	1100000013	7000000008	01-OCT-2023 12.00.00.000000000 AM
5	1400000034	Pending	(null)	(null)	15-AUG-2024	1100000014	7000000009	01-OCT-2024 12.00.00.000000000 AM
6	1400000001	Approved	01-FEB-2023	500	30-JAN-2023	1100000003	7000000004	09-DEC-2024 06.22.30.000000000 AM
7	1400000002	Denied	01-MAR-2023	0	25-FEB-2023	1100000001	7000000008	09-DEC-2024 06.22.30.000000000 AM
8	1400000003	Approved	15-MAR-2023	400	10-MAR-2023	1100000006	7000000002	09-DEC-2024 06.22.30.000000000 AM
9	1400000004	Pending	(null)	(null)	01-APR-2023	1100000008	7000000009	09-DEC-2024 06.22.30.000000000 AM
10	1400000005	Approved	20-MAY-2023	600	15-MAY-2023	1100000004	7000000007	09-DEC-2024 06.22.30.000000000 AM

After:

	APPROVAL_ID	APPROVAL_STATUS	APPROVAL_DATE	APPROVAL_AMOUNT	REVIEW_DATE	CLAIM_ID	EMPLOYEEID	LASTUPDATED
1	1400000030	Denied	(null)	(null)	01-JUL-2024	1100000010	7000000002	11-DEC-2024 07.35.26.000000000 PM
2	1400000031	Review	(null)	(null)	01-SEP-2024	1100000011	7000000004	11-DEC-2024 07.35.26.000000000 PM
3	1400000032	Closed	15-JUN-2023	3000	(null)	1100000012	7000000007	11-DEC-2024 07.35.26.000000000 PM
4	1400000033	Closed	20-MAY-2023	1500	(null)	1100000013	7000000008	11-DEC-2024 07.35.26.000000000 PM
5	1400000034	Review	(null)	(null)	15-AUG-2024	1100000014	7000000009	11-DEC-2024 07.35.26.000000000 PM
6	1400000001	Approved	01-FEB-2023	500	30-JAN-2023	1100000003	7000000004	09-DEC-2024 06.22.30.000000000 AM
7	1400000002	Denied	01-MAR-2023	0	25-FEB-2023	1100000001	7000000008	09-DEC-2024 06.22.30.000000000 AM
8	1400000003	Approved	15-MAR-2023	400	10-MAR-2023	1100000006	7000000002	09-DEC-2024 06.22.30.000000000 AM
9	1400000004	Pending	(null)	(null)	01-APR-2023	1100000008	7000000009	09-DEC-2024 06.22.30.000000000 AM
10	1400000005	Approved	20-MAY-2023	600	15-MAY-2023	1100000004	7000000007	09-DEC-2024 06.22.30.000000000 AM

2. 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:

	CONTRACT_ID	START_DATE	END_DATE	PLANID	RESELLER_ID	CUSTOMER_ID	DEALERID
1	1200000001	01-JAN-2023	01-JAN-2035	6000000003	1000000005	2000000003	3000000008
2	1200000002	01-FEB-2023	01-FEB-2033	6000000007	1000000002	2000000007	3000000003
3	1200000003	01-MAR-2023	01-MAR-2029	6000000002	1000000007	2000000005	3000000001
4	1200000004	01-APR-2023	01-APR-2030	6000000005	1000000009	2000000002	3000000007
5	1200000005	01-MAY-2023	01-MAY-2026	6000000004	1000000001	2000000004	3000000005
6	1200000006	01-JUN-2023	01-JUN-2032	6000000001	1000000004	2000000009	3000000002
7	1200000007	01-JUL-2023	01-JUL-2025	6000000006	1000000006	2000000006	3000000009
8	1200000008	01-AUG-2023	01-AUG-2027	6000000008	1000000003	2000000010	3000000004
9	1200000009	01-SEP-2023	01-SEP-2035	6000000003	1000000008	2000000008	3000000006
10	1200000010	01-OCT-2023	01-OCT-2033	6000000007	1000000010	2000000001	3000000003

3. Trigger 2: Claim Validation Rules

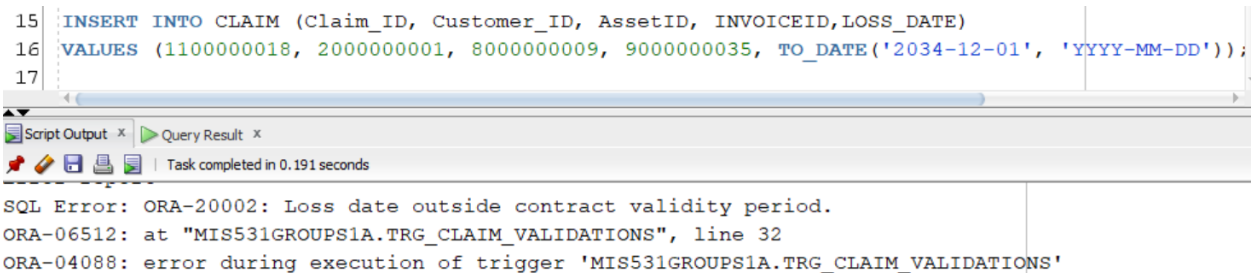
The trigger TRG_CLAIM_VALIDATIONS is designed to perform multiple validations before inserting or updating a claim in the CLAIM table. It checks the following conditions and raises errors if any of them are violated:

a. Check if the Loss Date is within the contract's validity period: (Loss date is the date the issue happened)
b. 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



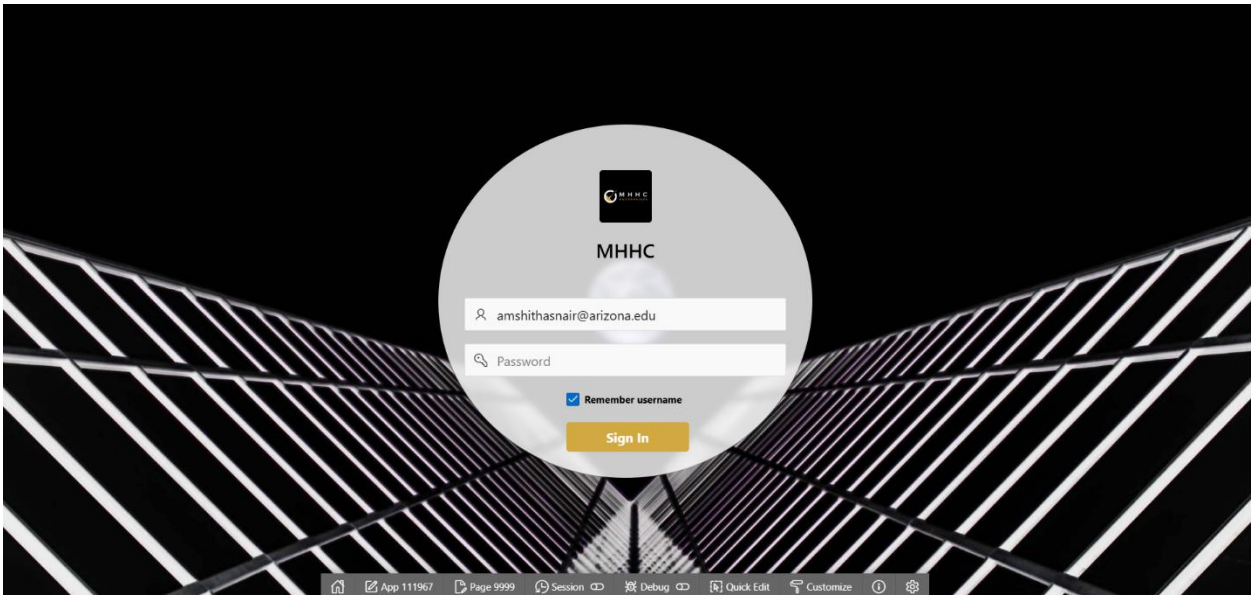
Error because Claim amount exceeds the asset cost. Asset cost is 1200

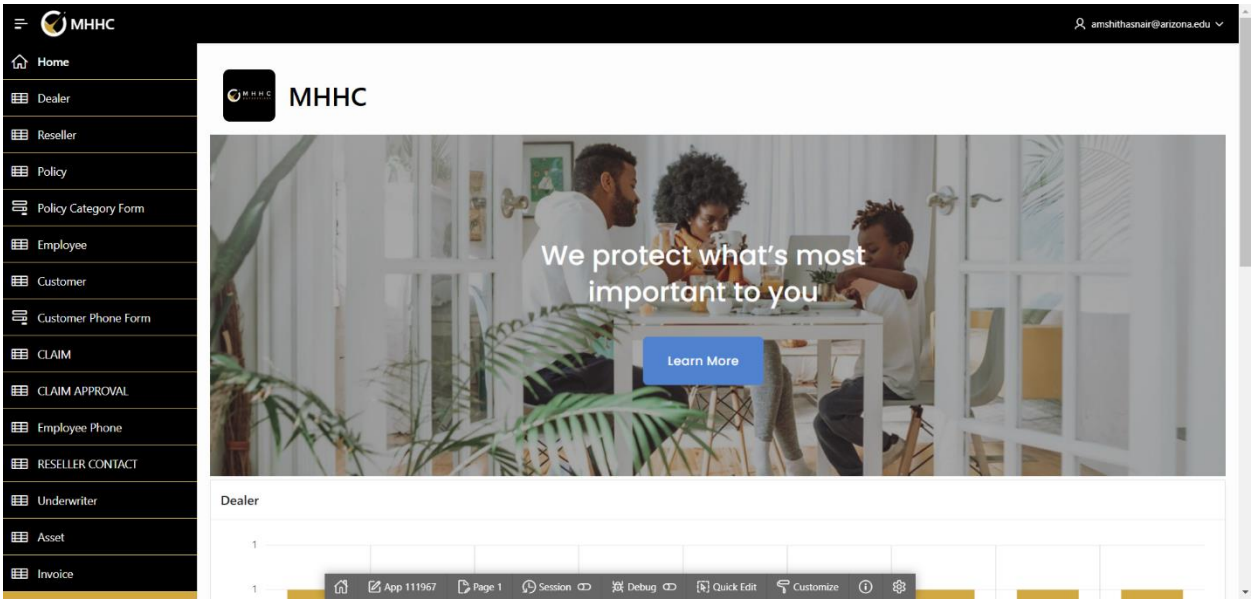
```
15  INSERT INTO CLAIM (Claim_ID, Customer_ID, AssetID, INVOICEID,LOSS_DATE,CLAIM_AMOUNT)
16  VALUES (1100000020, 2000000001, 8000000009, 9000000035, TO_DATE('2024-12-01', 'YYYY-MM-DD'),1400);
17
```

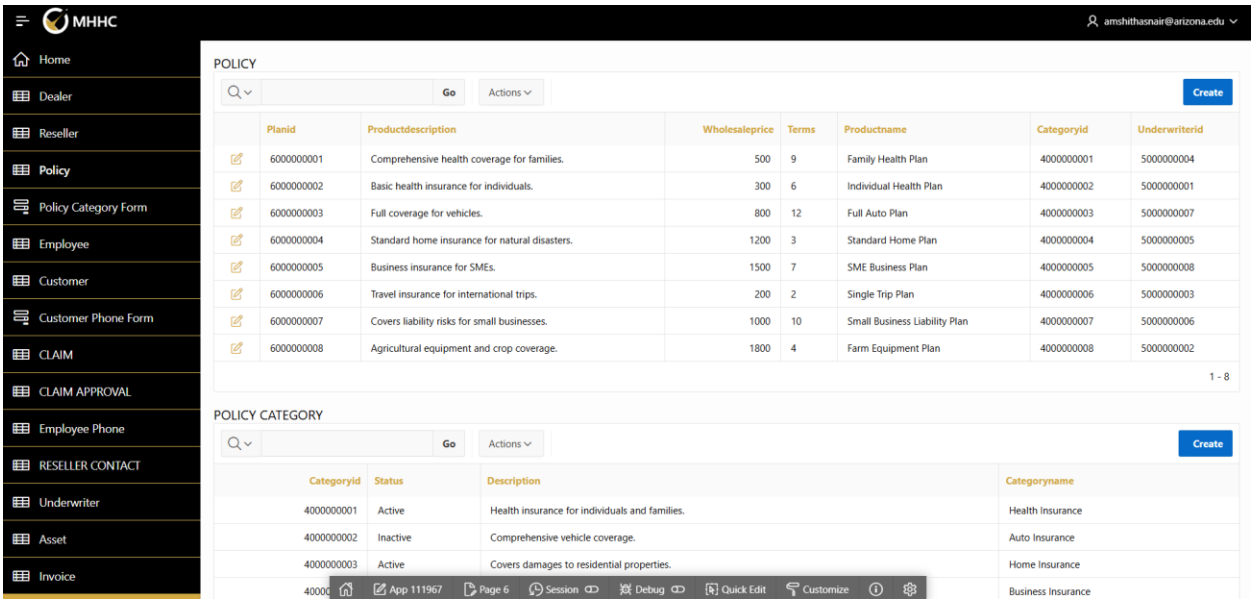
Script Output | Query Result | Task completed in 0.199 seconds

```
SQL Error: ORA-20003: Claim amount exceeds the asset cost.
ORA-06512: at "MIS531GROUPS1A.TRG_CLAIM_VALIDATIONS", line 43
ORA-04088: error during execution of trigger 'MIS531GROUPS1A.TRG_CLAIM_VALIDATIONS'
```

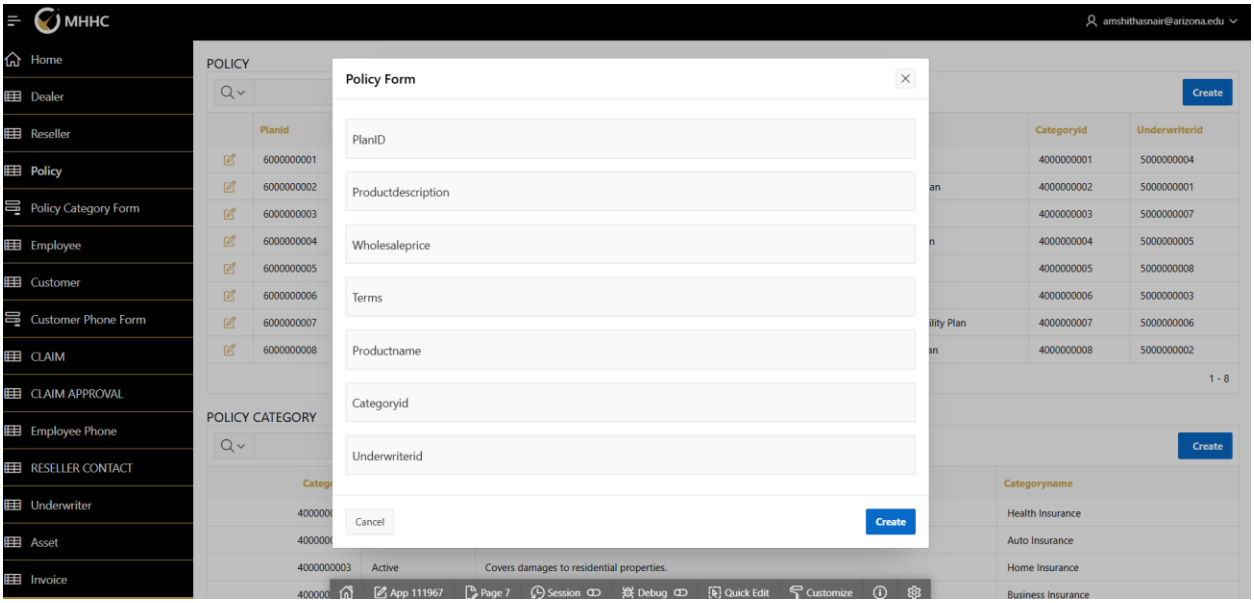
Chapter 6: Frontend Overview






MHHC
amshithasnair@arizona.edu
Home
Dealer
Reseller
Policy
Policy Category Form
Employee
Customer
Customer Phone Form
CLAIM
CLAIM APPROVAL
Employee Phone
RESELLER CONTACT
Underwriter
Asset
Invoice
POLICY
Go
Actions
Create
Planid
Productdescription
Wholesaleprice
Terms
Productname
Categoryid
Underwriterid
6000000001 Comprehensive health coverage for families. 500 9 Family Health Plan 4000000001 5000000004
6000000002 Basic health insurance for individuals. 300 6 Individual Health Plan 4000000002 5000000001
6000000003 Full coverage for vehicles. 800 12 Full Auto Plan 4000000003 5000000007
6000000004 Standard home insurance for natural disasters. 1200 3 Standard Home Plan 4000000004 5000000005
6000000005 Business insurance for SMEs. 1500 7 SME Business Plan 4000000005 5000000008
6000000006 Travel insurance for international trips. 200 2 Single Trip Plan 4000000006 5000000003
6000000007 Covers liability risks for small businesses. 1000 10 Small Business Liability Plan 4000000007 5000000006
6000000008 Agricultural equipment and crop coverage. 1800 4 Farm Equipment Plan 4000000008 5000000002
1 - 8
POLICY CATEGORY
Go
Actions
Create
Categoryid
Status
Description
Categoryname
4000000001 Active Health insurance for individuals and families. Health Insurance
4000000002 Inactive Comprehensive vehicle coverage. Auto Insurance
4000000003 Active Covers damages to residential properties. Home Insurance
Business Insurance
App 111967
Page 6
Session
Debug
Quick Edit
Customize

dry


MHHC
amshithasnair@arizona.edu
Home
Dealer
Reseller
Policy
Policy Category Form
Employee
Customer
Customer Phone Form
CLAIM
CLAIM APPROVAL
Employee Phone
RESELLER CONTACT
Underwriter
Asset
Invoice
POLICY
Policy Form
PlanID
Productdescription
Wholesaleprice
Terms
Productname
Categoryid
Underwriterid
Cancel
Create
POLICY CATEGORY
Create
Health Insurance
Auto Insurance
4000000003 Active Covers damages to residential properties. Home Insurance
Business Insurance
App 111967
Page 7
Session
Debug
Quick Edit
Customize

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.

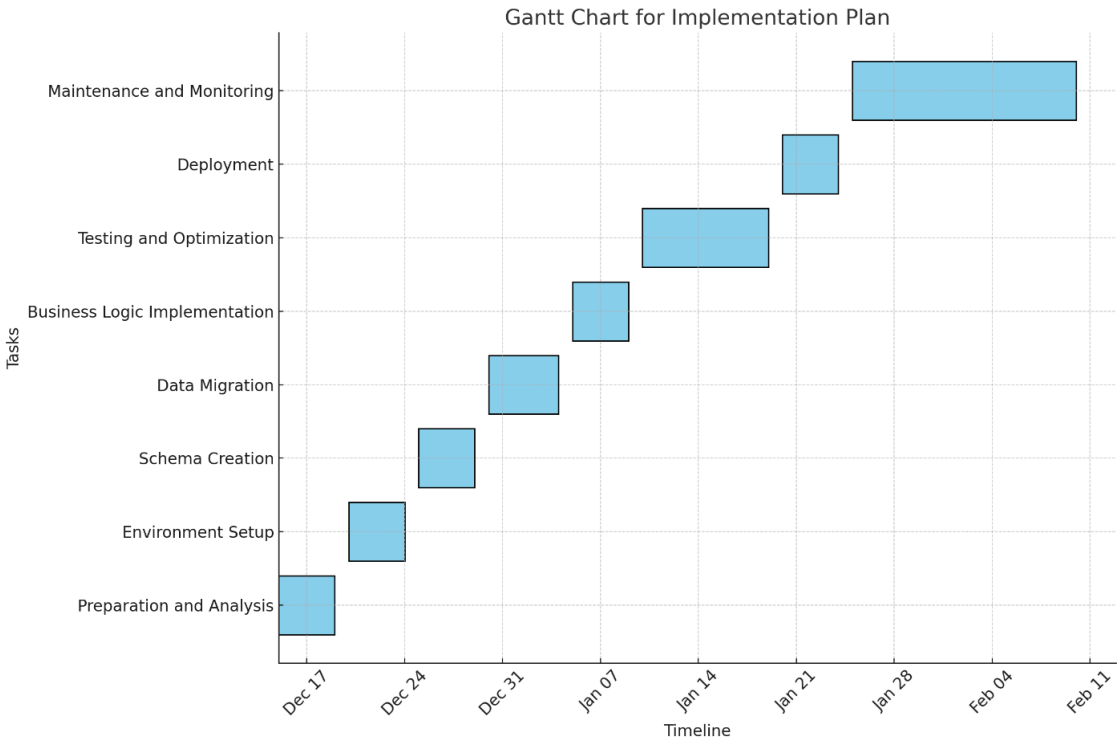
Step 8: Maintenance and Monitoring

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.
2. **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.
3. **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.
4. **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.
5. **Learnings from Other Teams:**
   - Observing other teams' presentations emphasized the value of well-documented processes and visual aids to communicate complex ideas effectively.
6. **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.