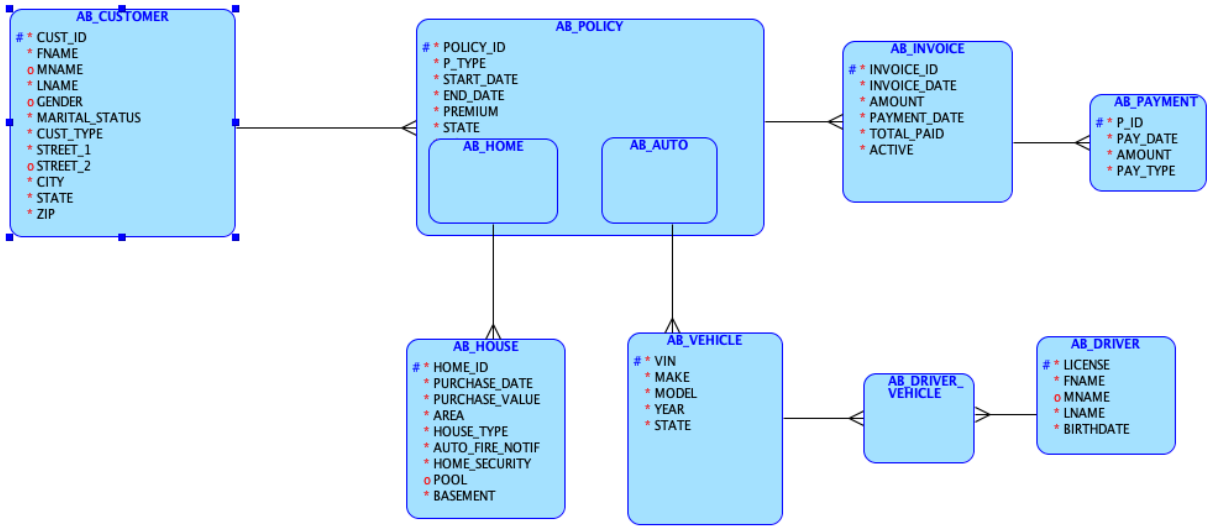
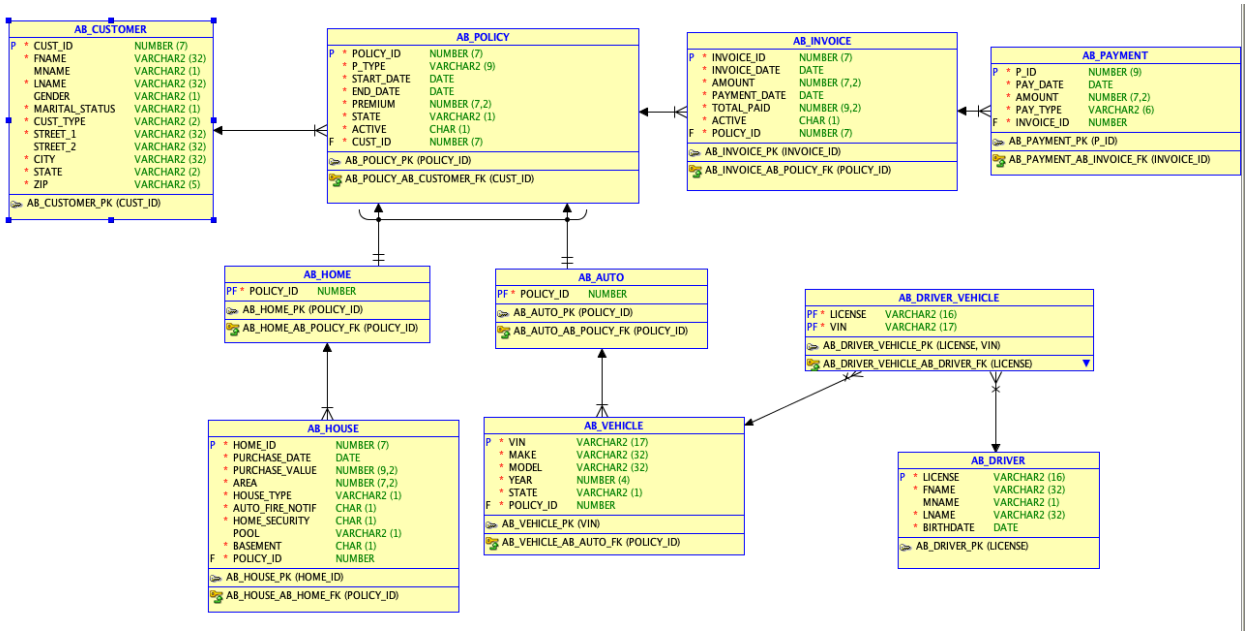
Project_Part_1_CSGY6083A_ab7289

Course: CSGY-6083
Section: A
Submission Date: 2021-07-30
Student Name: Alex Biehl
Student ID: N12370097

Logical Model



Relational Model



DDL Script

Below is the DDL script, first generated by OracleDataModeler, then converted from Oracle SQl to MySQL using sqlines.com and some manual conversion.

START SCRIPT

```
DROP DATABASE IF EXISTS ab_project;
CREATE DATABASE IF NOT EXISTS ab_project;

USE ab_project;

DROP TABLE IF EXISTS ab_auto;
CREATE TABLE ab_auto (
    policy_id INT NOT NULL COMMENT 'THE INSURANCE POLICY UNIQUE ID'
);

ALTER TABLE ab_auto ADD CONSTRAINT ab_auto_pk PRIMARY KEY ( policy_id );

ALTER TABLE ab_auto MODIFY COLUMN policy_id INT AUTO_INCREMENT UNIQUE;

USE ab_project;

DROP TABLE IF EXISTS ab_customer;
CREATE TABLE ab_customer (
    cust_id         INT NOT NULL COMMENT 'THE CUSTOMERS UNIQUE ID',
    fname           VARCHAR(32) NOT NULL COMMENT 'THE CUSTOMERS FIRST NAME',
    mname            VARCHAR(1) COMMENT 'THE CUSTOMERS MIDDLE INITIAL',
    lname           VARCHAR(32) NOT NULL COMMENT 'THE CUSTOMERS LAST NAME',
    gender          VARCHAR(1) COMMENT 'THE CUSTOMERS GENDER',
    marital_status  VARCHAR(1) NOT NULL COMMENT 'THE CUSTOMERS MARITAL
STATUS. EITHER "M", "S", OR "W"',
    cust_type       VARCHAR(2) NOT NULL COMMENT 'THE CUSTOMER TYPE. "A"
AUTOMOBILE INSURANCE, "H" HOME INSURANCE.',
    street_1        VARCHAR(32) NOT NULL COMMENT 'CUSTOMERS STREET ADDRESS',
    street_2        VARCHAR(32) COMMENT 'OPTIONAL CUSTOMER FLOOR/APARTMENT.',
    city            VARCHAR(32) NOT NULL COMMENT 'THE CUSTOMERS ADDRESS CITY.',
    state           VARCHAR(2) NOT NULL COMMENT 'THE CUSTOMERS HOME ADDRESS
STATE CODE',
    zip             VARCHAR(5) NOT NULL COMMENT 'THE CUSTOMERS HOME ADDRESS
ZIPCODE.'
);
```

```
ALTER TABLE ab_customer ADD CONSTRAINT ab_customer_pk PRIMARY KEY ( cust_id );

ALTER TABLE ab_customer MODIFY COLUMN cust_id INT AUTO_INCREMENT UNIQUE;

USE ab_project;

DROP TABLE IF EXISTS ab_driver;
CREATE TABLE ab_driver (
   license    VARCHAR(16) NOT NULL COMMENT 'THE DRIVERS LICENSE NUMBER',
   fname      VARCHAR(32) NOT NULL COMMENT 'THE DRIVERS FIRST NAME.',
   mname      VARCHAR(1) COMMENT 'THE DRIVERS OPTIONAL MIDDLE INITIAL.',
   lname      VARCHAR(32) NOT NULL COMMENT 'THE DRIVERS LAST NAME.',
   birthdate  DATETIME NOT NULL COMMENT 'THE DRIVERS BIRTHDATE.'
);


ALTER TABLE ab_driver ADD CONSTRAINT ab_driver_pk PRIMARY KEY ( license );

ALTER TABLE ab_driver MODIFY COLUMN license VARCHAR(16) UNIQUE;

USE ab_project;

DROP TABLE IF EXISTS ab_driver_vehicle;
CREATE TABLE ab_driver_vehicle (
   license  VARCHAR(16) NOT NULL COMMENT 'THE LICENSE OF THE CARS DRIVER',
   vin      VARCHAR(17) NOT NULL COMMENT 'THE VIN OF THE INSURED VEHICLE'
);

ALTER TABLE ab_driver_vehicle ADD CONSTRAINT ab_driver_vehicle_pk PRIMARY KEY (
license,
                                                vin );

USE ab_project;

DROP TABLE IF EXISTS ab_home;
CREATE TABLE ab_home (
   policy_id INT NOT NULL COMMENT 'THE INSURANCE POLICY UNIQUE ID'
);

ALTER TABLE ab_home ADD CONSTRAINT ab_home_pk PRIMARY KEY ( policy_id );

USE ab_project;

DROP TABLE IF EXISTS ab_house;
```

```
CREATE TABLE ab_house (
   home_id          INT NOT NULL COMMENT 'THE UNIQUE HOME ID',
   purchase_date    DATETIME NOT NULL COMMENT 'THE DATE THE HOME WAS
PURCHASED.',
   purchase_value   DECIMAL(9, 2) NOT NULL COMMENT 'THE HOMES PURCHASE
VALUE.',
   area             DECIMAL(7, 2) NOT NULL COMMENT 'THE HOMES AREA IN SQUARE
FEET.',
   house_type             VARCHAR(1) NOT NULL COMMENT 'THE HOME TYPE. "S" IS SINGLE
FAMILY, "M" IS MULTI FAMILY, "C" IS CONDOMINIUM, "T" IS TOWN HOUSE.',
   auto_fire_notif  TINYINT NOT NULL COMMENT 'WHETHER THE HOUSE HAS
AUTOMATIC FIRE NOTIFICATION TO THE FIRE DEPARTMENT.',
   home_security    TINYINT NOT NULL COMMENT 'WHETHER THE HOUSE HAS A
SECURITY SYSTEM.',
   pool             VARCHAR(1) COMMENT 'SWIMMING POOL. "U" IS UNDERGROUND, "O" IS
OVERGROUND, "I" IS INDOOR, "M" IS MULTIPLE, NULL IS NO POOL.',
   basement         TINYINT NOT NULL COMMENT 'WHETHER THE HOUSE HAS A
BASEMENT.',
   policy_id        INT NOT NULL COMMENT 'THE ID OF THE POLICY INSURING THE
HOUSE'
);

ALTER TABLE ab_house ADD CONSTRAINT ab_house_pk PRIMARY KEY ( home_id );

ALTER TABLE ab_house MODIFY COLUMN home_id INT AUTO_INCREMENT UNIQUE;

USE ab_project;

DROP TABLE IF EXISTS ab_invoice;
CREATE TABLE ab_invoice (
   invoice_id    INT NOT NULL COMMENT 'THE INVOICE ID',
   invoice_date  DATETIME NOT NULL COMMENT 'THE DATE GENERATED.',
   amount        DECIMAL(7, 2) NOT NULL COMMENT 'THE AMOUNT DUE',
   payment_date  DATETIME NOT NULL COMMENT 'THE DATE THE INVOICE IS DUE.',
   total_paid    DECIMAL(9, 2) NOT NULL COMMENT 'The amount that the client has paid so
far.',
   active        TINYINT NOT NULL COMMENT 'WHETHER THE INVOICE IS ACTIVE',
   policy_id     INT NOT NULL COMMENT 'ID OF THE POLICY THAT THE INVOICE
BELONGS TO'
);

ALTER TABLE ab_invoice ADD CONSTRAINT ab_invoice_pk PRIMARY KEY ( invoice_id );

ALTER TABLE ab_invoice MODIFY COLUMN invoice_id INT AUTO_INCREMENT UNIQUE;
```

```
USE ab_project;

DROP TABLE IF EXISTS ab_payment;
CREATE TABLE ab_payment (
   p_id        BIGINT NOT NULL COMMENT 'THE PAYMENT ID',
   pay_date    DATETIME NOT NULL COMMENT 'THE DATE THE PAYMENT WAS MADE',
   amount      DECIMAL(7, 2) NOT NULL COMMENT 'THE PAYMENT INSTALLMENT
AMOUNT.',
   pay_type        VARCHAR(6) NOT NULL COMMENT 'THE METHOD OF PAYMENT; ONE OF
"PayPal", "Credit", "Debit", OR "Check".',
   invoice_id  INT NOT NULL COMMENT 'THE INVOICE THE PAYMENT IS GOING
TOWARDS'
);


ALTER TABLE ab_payment ADD CONSTRAINT ab_payment_pk PRIMARY KEY ( p_id );

ALTER TABLE ab_payment MODIFY COLUMN p_id BIGINT AUTO_INCREMENT UNIQUE;

USE ab_project;

DROP TABLE IF EXISTS ab_policy;
CREATE TABLE ab_policy (
   policy_id   INT NOT NULL COMMENT 'THE INSURANCE POLICY UNIQUE ID',
   p_type        VARCHAR(9) NOT NULL COMMENT 'THE POLICY TYPE. "A" FOR AUTO AND
"H" FOR HOME.',
   start_date  DATETIME NOT NULL COMMENT 'THE POLICY START DATE',
   end_date    DATETIME NOT NULL COMMENT 'THE POLICY END DATE.',
   premium     DECIMAL(7, 2) NOT NULL COMMENT 'THE PREMIUM AMOUNT.',
   state      VARCHAR(1) NOT NULL COMMENT 'THE POLICY STATUS. "C" FOR CURRENT,
"P" FOR EXPIRED.',
   active      TINYINT NOT NULL COMMENT 'WHETHER THE POLICY IS STILL ACTIVE',
   cust_id     INT NOT NULL COMMENT 'ID OF THE CUSTOMER HOLDING THE POLICY'
);

ALTER TABLE ab_policy
   ADD CONSTRAINT ch_inh_ab_policy CHECK ( p_type IN ( 'A', 'H', 'AH' ) );

ALTER TABLE ab_policy ADD CONSTRAINT ab_policy_pk PRIMARY KEY ( policy_id );

ALTER TABLE ab_policy MODIFY COLUMN policy_id INT AUTO_INCREMENT UNIQUE;

USE ab_project;
```

```
DROP TABLE IF EXISTS ab_vehicle;
CREATE TABLE ab_vehicle (
   vin        VARCHAR(17) NOT NULL COMMENT 'THE VEHICLE IDENTIFICATION NUMBER',
   make       VARCHAR(32) NOT NULL COMMENT 'THE VEHICLE MAKE.',
   model      VARCHAR(32) NOT NULL COMMENT 'THE VEHICLE MODEL.',
   year       SMALLINT NOT NULL COMMENT 'THE VEHICLE YEAR.',
   state      VARCHAR(1) NOT NULL COMMENT 'VEHICLE STATUS. "L" IS LEASED, "F" IS
FINANCED, AND "O" IS OWNED.',
   policy_id  INT NOT NULL COMMENT 'THE ID OF THE POLICY INSURING THE CAR'
);


ALTER TABLE ab_vehicle ADD CONSTRAINT ab_vehicle_pk PRIMARY KEY ( vin );

ALTER TABLE ab_vehicle MODIFY COLUMN vin VARCHAR(17)  UNIQUE;

ALTER TABLE ab_auto
   ADD CONSTRAINT ab_auto_ab_policy_fk FOREIGN KEY ( policy_id )
      REFERENCES ab_policy ( policy_id );

ALTER TABLE ab_driver_vehicle
   ADD CONSTRAINT ab_driver_ab_vehicle_fk FOREIGN KEY ( vin )
      REFERENCES ab_vehicle ( vin )
         ON DELETE CASCADE;

ALTER TABLE ab_driver_vehicle
   ADD CONSTRAINT ab_driver_vehicle_ab_driver_fk FOREIGN KEY ( license )
      REFERENCES ab_driver ( license )
         ON DELETE CASCADE;

ALTER TABLE ab_home
   ADD CONSTRAINT ab_home_ab_policy_fk FOREIGN KEY ( policy_id )
      REFERENCES ab_policy ( policy_id );

ALTER TABLE ab_house
   ADD CONSTRAINT ab_house_ab_home_fk FOREIGN KEY ( policy_id )
      REFERENCES ab_home ( policy_id );

ALTER TABLE ab_invoice
   ADD CONSTRAINT ab_invoice_ab_policy_fk FOREIGN KEY ( policy_id )
      REFERENCES ab_policy ( policy_id );

ALTER TABLE ab_payment
```

```
   ADD CONSTRAINT ab_payment_ab_invoice_fk FOREIGN KEY ( invoice_id )
      REFERENCES ab_invoice ( invoice_id );

ALTER TABLE ab_policy
   ADD CONSTRAINT ab_policy_ab_customer_fk FOREIGN KEY ( cust_id )
      REFERENCES ab_customer ( cust_id );

ALTER TABLE ab_vehicle
   ADD CONSTRAINT ab_vehicle_ab_auto_fk FOREIGN KEY ( policy_id )
      REFERENCES ab_auto ( policy_id );


-- custom constraints

ALTER TABLE ab_invoice ALTER total_paid SET DEFAULT 0;

ALTER TABLE ab_customer
        ADD CONSTRAINT c_customer_gender
                CHECK (gender IN ('M', 'F', NULL));

ALTER TABLE ab_customer
        ADD CONSTRAINT c_customer_marry
                CHECK (marital_status IN ('M', 'S', 'W'));

ALTER TABLE ab_customer
        ADD CONSTRAINT c_customer_type
                CHECK (cust_type IN ('A', 'H', 'AH'));

ALTER TABLE ab_policy
        ADD CONSTRAINT c_policy_status
                CHECK (state IN ('C', 'P'));

ALTER TABLE ab_policy
        ADD CONSTRAINT c_policy_active
                CHECK (active IN (1, 0));

ALTER TABLE ab_house
        ADD CONSTRAINT c_home_house_type
                CHECK (house_type IN ('S', 'M', 'C', 'T'));

ALTER TABLE ab_house
        ADD CONSTRAINT c_home_fire_notif
                CHECK (auto_fire_notif IN (0, 1));
```

```
ALTER TABLE ab_house
	ADD CONSTRAINT c_home_sec_sys
		CHECK (home_security IN (0, 1));

ALTER TABLE ab_house
	ADD CONSTRAINT c_home_pool
		CHECK (pool IN ('U', 'O', 'I', 'M', NULL));

ALTER TABLE ab_house
	ADD CONSTRAINT c_home_basement
		CHECK (basement IN (0, 1));

ALTER TABLE ab_payment
	ADD CONSTRAINT c_payment_type
		CHECK (pay_type IN ('PayPal', 'Credit', 'Debit', 'Check'));

ALTER TABLE ab_vehicle
	ADD CONSTRAINT c_vehicle_status
		CHECK (state IN ('L', 'F', 'O'));

-- SQLINES DEMO *** aints

delimiter |

DROP TRIGGER IF EXISTS arc_fkarc_2_ab_home |
CREATE TRIGGER arc_fkarc_2_ab_home BEFORE
   INSERT ON ab_home
   FOR EACH ROW
BEGIN
   DECLARE d VARCHAR(9);
   -- SQLINES LICENSE FOR EVALUATION USE ONLY
   SELECT
      a.p_type
   INTO d
   FROM
      ab_policy a
   WHERE
      a.policy_id = new.policy_id;

   IF  d IS NULL OR d <> 'H' THEN
                        -- set msg = 'FK AB_HOME_AB_POLICY_FK in Table AB_HOME violates
Arc constraint on Table AB_POLICY - discriminator column TYPE doesn''t have value ''H''';
                        signal sqlstate '45000'
```

```
                                    SET message_text = 'Cannot associate a home with an insurance
policy without type "H"';
   END if ;

   -- DECLARE EXIT HANDLER FOR not found BEGIN
      -- NULL;
   -- END;
   -- DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN
      -- RESIGNAL;
   -- END;
END
|
delimiter ;

DELIMITER |

DROP TRIGGER IF EXISTS arc_fkarc_2_ab_home |
CREATE TRIGGER arc_fkarc_2_ab_home BEFORE
   UPDATE ON ab_home
   FOR EACH ROW
BEGIN
   DECLARE d VARCHAR(9);
   -- SQLINES LICENSE FOR EVALUATION USE ONLY
   SELECT
      a.p_type
   INTO d
   FROM
      ab_policy a
   WHERE
      a.policy_id = new.policy_id;

   IF ( d IS NULL OR d <> 'H' ) THEN
                  SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'Cannot associate a home with an insurance
policy without type "H"';
   END IF;

   -- DECLARE EXIT HANDLER FOR not found BEGIN
      -- NULL;
   -- END;
   -- DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN
      -- RESIGNAL;
   -- END;
END
```

```
|
DELIMITER ;

DELIMITER |

DROP TRIGGER IF EXISTS arc_fkarc_2_ab_auto |
CREATE TRIGGER arc_fkarc_2_ab_auto BEFORE
   INSERT ON ab_auto
   FOR EACH ROW
BEGIN
   DECLARE d VARCHAR(9);
   -- SQLINES LICENSE FOR EVALUATION USE ONLY
   SELECT
      a.p_type
   INTO d
   FROM
      ab_policy a
   WHERE
      a.policy_id = new.policy_id;

   IF ( d IS NULL OR d <> 'A' ) THEN
                  SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'Cannot associate a vehilce with an insurance
policy without type ''A''';
   END IF;

   -- DECLARE EXIT HANDLER FOR not found BEGIN
      -- NULL;
   -- END;
   -- DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN
      -- RESIGNAL;
   -- END;
END
 |

DELIMITER ;

DELIMITER |

DROP TRIGGER IF EXISTS arc_fkarc_2_ab_auto |
CREATE TRIGGER arc_fkarc_2_ab_auto BEFORE
   UPDATE ON ab_auto
   FOR EACH ROW
BEGIN
```

```
    DECLARE d VARCHAR(9);
    -- SQLINES LICENSE FOR EVALUATION USE ONLY
    SELECT
        a.type
    INTO d
    FROM
        ab_policy a
    WHERE
        a.policy_id = new.policy_id;

    IF ( d IS NULL OR d <> 'A' ) THEN
                    SIGNAL SQLSTATE '45000'
                            SET MESSAGE_TEXT = 'Cannot associate a vehicle with an insurance
policy without type "A"';
    END IF;

    -- DECLARE EXIT HANDLER FOR not found BEGIN
        -- NULL;
    -- END;
    -- DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN
        -- RESIGNAL;
    -- END;
END
|

DELIMITER ;

DELIMITER |

DROP TRIGGER IF EXISTS tr_policy_insert_end_date |
CREATE TRIGGER tr_policy_insert_end_date BEFORE
            INSERT ON ab_policy
    FOR EACH ROW
BEGIN
            DECLARE d DATETIME;
    SELECT
                    a.start_date
            INTO d
    FROM
                    ab_policy a
            WHERE
                    a.policy_id = NEW.policy_id;

    IF (d IS NOT NULL AND d > NEW.end_date) THEN
```

```
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: The end date cannot be before the
start date';
        END IF;
END
|

DROP TRIGGER IF EXISTS tr_policy_update_end_date |
CREATE TRIGGER tr_policy_update_end_date BEFORE
        UPDATE ON ab_policy
   FOR EACH ROW
BEGIN
        DECLARE d DATETIME;
   SELECT
                a.start_date
        INTO d
   FROM
                ab_policy a
        WHERE
                a.policy_id = NEW.policy_id;

   IF (d IS NOT NULL AND d > NEW.end_date) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: The end date cannot be before the
start date';
        END IF;
END
|

DROP TRIGGER IF EXISTS tr_policy_insert_start_date |
CREATE TRIGGER tr_policy_insert_start_date BEFORE
        INSERT ON ab_policy
   FOR EACH ROW
BEGIN
        DECLARE d DATETIME;
   SELECT
                a.end_date
        INTO d
   FROM
                ab_policy a
        WHERE
                a.policy_id = NEW.policy_id;

        IF (d IS NOT NULL AND d < NEW.start_date) THEN
```

```
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: The Start date cannot be after the
end date';
        END IF;
END
|

DROP TRIGGER IF EXISTS tr_policy_update_start_date |
CREATE TRIGGER tr_policy_update_start_date BEFORE
        UPDATE ON ab_policy
   FOR EACH ROW
BEGIN
        DECLARE d DATETIME;
   SELECT
                a.end_date
        INTO d
   FROM
                ab_policy a
        WHERE
                a.policy_id = NEW.policy_id;

        IF (d IS NOT NULL AND d < NEW.start_date) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: The Start date cannot be after the
end date';
        END IF;
END
|

-- trigger to insure that AB_POLICY.PREMIUM is not negative

DROP TRIGGER IF EXISTS tr_policy_ins_prem |
CREATE TRIGGER tr_policy_ins_prem BEFORE
        INSERT ON ab_policy
   FOR EACH ROW
BEGIN
        IF (NEW.premium IS NULL OR NEW.premium < 0) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: Cannot create a policy with a
negative premium';
        END IF;
END
 |
```

```
DROP TRIGGER IF EXISTS tr_policy_upd_prem |
CREATE TRIGGER tr_policy_upd_prem BEFORE
        UPDATE ON ab_policy
   FOR EACH ROW
BEGIN
        IF (NEW.premium IS NULL OR NEW.premium <= 0) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: Cannot create a policy with a
negative premium';
        END IF;
END
 |

-- trigger to insure that AB_PAYMENT.AMOUNT is not negative

DROP TRIGGER IF EXISTS tr_pay_ins_amnt |
CREATE TRIGGER tr_pay_ins_amnt BEFORE
        INSERT ON ab_payment
   FOR EACH ROW
BEGIN
        IF (NEW.amount IS NULL OR NEW.amount <= 0) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: Cannot create a payment that is 0 or
negative';
        END IF;
END

|

-- Trigger to deduct a payment from the total owed on an invoice

DROP TRIGGER IF EXISTS tr_pay_ins_updtotal |
CREATE TRIGGER tr_pay_ins_updtotal BEFORE
        INSERT ON ab_payment
   FOR EACH ROW FOLLOWS tr_pay_ins_amnt
BEGIN
        UPDATE ab_invoice
   SET total_paid = total_paid + NEW.amount
   WHERE invoice_id = NEW.invoice_id;
END

|

-- Trigger to deactivate an invoice if it has been paid off fully
```

```
DROP TRIGGER IF EXISTS tr_invoice_upd_deactivate |
CREATE TRIGGER tr_invoice_upd_deactivate BEFORE INSERT ON ab_invoice
FOR EACH ROW
BEGIN
	DECLARE d DECIMAL(9,2);
   DECLARE a DECIMAL(7,2);
	SELECT
		a.total_paid, a.amount
	INTO d, a
	FROM
		ab_invoice a
	WHERE
		a.invoice_id = NEW.invoice_id;

	IF (NEW.active <> 0 and d IS NOT NULL AND d <> NEW.total_paid AND
NEW.total_paid >= a) THEN
		SET NEW.active = 0;
	END IF;
END

|



DELIMITER ;


-- SQLINES DEMO *** per Data Modeler Summary Report:
--
-- SQLINES DEMO ***                    10
-- SQLINES DEMO ***                     0
-- SQLINES DEMO ***                    20
-- SQLINES DEMO ***                     0
-- SQLINES DEMO ***                     0
-- SQLINES DEMO ***                     0
-- SQLINES DEMO *** DY                    0
-- SQLINES DEMO ***                     0
-- SQLINES DEMO ***                     0
-- SQLINES DEMO ***                     2
-- SQLINES DEMO ***                     0
-- SQLINES DEMO ***  TYPE                 0
-- SQLINES DEMO ***  TYPE                 0
-- SQLINES DEMO ***  TYPE BODY            0
```

```
-- SQLINES DEMO ***                    0
-- SQLINES DEMO ***                    0
-- SQLINES DEMO ***                    0
-- SQLINES DEMO ***                    0
-- SQLINES DEMO ***                    0
-- SQLINES DEMO ***                    0
-- SQLINES DEMO ***                    0
-- SQLINES DEMO *** EGMENT              0
-- SQLINES DEMO ***                    0
-- SQLINES DEMO *** ED VIEW             0
-- SQLINES DEMO *** ED VIEW LOG          0
-- SQLINES DEMO ***                    0
-- SQLINES DEMO ***                    0
-- SQLINES DEMO ***                    0
--
-- SQLINES DEMO ***                    0
-- SQLINES DEMO ***                    0
--
-- SQLINES DEMO ***                    0
--
-- SQLINES DEMO ***                    0
-- SQLINES DEMO *** A                   0
-- SQLINES DEMO *** T                   0
--
-- SQLINES DEMO ***                    0
-- SQLINES DEMO ***                    0
```

Constraints and Triggers

These constraints were added to the tables to ensure data consistency.

```
ALTER TABLE ab_invoice ALTER total_paid SET DEFAULT 0;

ALTER TABLE ab_customer
        ADD CONSTRAINT c_customer_gender
                CHECK (gender IN ('M', 'F', NULL));

ALTER TABLE ab_customer
        ADD CONSTRAINT c_customer_marry
                CHECK (marital_status IN ('M', 'S', 'W'));

ALTER TABLE ab_customer
```

```
        ADD CONSTRAINT c_customer_type
                CHECK (cust_type IN ('A', 'H', 'AH'));

ALTER TABLE ab_policy
        ADD CONSTRAINT c_policy_status
                CHECK (state IN ('C', 'P'));

ALTER TABLE ab_policy
        ADD CONSTRAINT c_policy_active
                CHECK (active IN (1, 0));

ALTER TABLE ab_house
        ADD CONSTRAINT c_home_house_type
                CHECK (house_type IN ('S', 'M', 'C', 'T'));

ALTER TABLE ab_house
        ADD CONSTRAINT c_home_fire_notif
                CHECK (auto_fire_notif IN (0, 1));

ALTER TABLE ab_house
        ADD CONSTRAINT c_home_sec_sys
                CHECK (home_security IN (0, 1));

ALTER TABLE ab_house
        ADD CONSTRAINT c_home_pool
                CHECK (pool IN ('U', 'O', 'I', 'M', NULL));

ALTER TABLE ab_house
        ADD CONSTRAINT c_home_basement
                CHECK (basement IN (0, 1));

ALTER TABLE ab_payment
        ADD CONSTRAINT c_payment_type
                CHECK (pay_type IN ('PayPal', 'Credit', 'Debit', 'Check'));

ALTER TABLE ab_vehicle
        ADD CONSTRAINT c_vehicle_status
                CHECK (state IN ('L', 'F', 'O'));

-- SQLINES DEMO *** aints

delimiter |

DROP TRIGGER IF EXISTS arc_fkarc_2_ab_home |
```

```
CREATE TRIGGER arc_fkarc_2_ab_home BEFORE
   INSERT ON ab_home
   FOR EACH ROW
BEGIN
   DECLARE d VARCHAR(9);
   -- SQLINES LICENSE FOR EVALUATION USE ONLY
   SELECT
      a.p_type
   INTO d
   FROM
      ab_policy a
   WHERE
      a.policy_id = new.policy_id;

   IF  d IS NULL OR d <> 'H' THEN
                        -- set msg = 'FK AB_HOME_AB_POLICY_FK in Table AB_HOME violates
Arc constraint on Table AB_POLICY - discriminator column TYPE doesn''t have value ''H''';
                        signal sqlstate '45000'
                                SET message_text = 'Cannot associate a home with an insurance
policy without type ''H''';
   END if ;

   -- DECLARE EXIT HANDLER FOR not found BEGIN
      -- NULL;
   -- END;
   -- DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN
      -- RESIGNAL;
   -- END;
END
|
delimiter ;

DELIMITER |

DROP TRIGGER IF EXISTS arc_fkarc_2_ab_home |
CREATE TRIGGER arc_fkarc_2_ab_home BEFORE
   UPDATE ON ab_home
   FOR EACH ROW
BEGIN
   DECLARE d VARCHAR(9);
   -- SQLINES LICENSE FOR EVALUATION USE ONLY
   SELECT
      a.p_type
   INTO d
```

```
    FROM
        ab_policy a
    WHERE
        a.policy_id = new.policy_id;

    IF ( d IS NULL OR d <> 'H' ) THEN
                    SIGNAL SQLSTATE '45000'
                            SET MESSAGE_TEXT = 'Cannot associate a home with an insurance
policy without type "H"';
    END IF;

    -- DECLARE EXIT HANDLER FOR not found BEGIN
        -- NULL;
    -- END;
    -- DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN
        -- RESIGNAL;
    -- END;
END
|
DELIMITER ;

DELIMITER |

DROP TRIGGER IF EXISTS arc_fkarc_2_ab_auto |
CREATE TRIGGER arc_fkarc_2_ab_auto BEFORE
    INSERT ON ab_auto
    FOR EACH ROW
BEGIN
    DECLARE d VARCHAR(9);
    -- SQLINES LICENSE FOR EVALUATION USE ONLY
    SELECT
        a.p_type
    INTO d
    FROM
        ab_policy a
    WHERE
        a.policy_id = new.policy_id;

    IF ( d IS NULL OR d <> 'A' ) THEN
                    SIGNAL SQLSTATE '45000'
                            SET MESSAGE_TEXT = 'Cannot associate a vehilce with an insurance
policy without type "A"';
    END IF;
```

```
   -- DECLARE EXIT HANDLER FOR not found BEGIN
      -- NULL;
   -- END;
   -- DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN
      -- RESIGNAL;
   -- END;
END
 |

DELIMITER ;

DELIMITER |

DROP TRIGGER IF EXISTS arc_fkarc_2_ab_auto |
CREATE TRIGGER arc_fkarc_2_ab_auto BEFORE
   UPDATE ON ab_auto
   FOR EACH ROW
BEGIN
   DECLARE d VARCHAR(9);
   -- SQLINES LICENSE FOR EVALUATION USE ONLY
   SELECT
      a.type
   INTO d
   FROM
      ab_policy a
   WHERE
      a.policy_id = new.policy_id;

   IF ( d IS NULL OR d <> 'A' ) THEN
                  SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'Cannot associate a vehicle with an insurance
policy without type "A"';
   END IF;

   -- DECLARE EXIT HANDLER FOR not found BEGIN
      -- NULL;
   -- END;
   -- DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN
      -- RESIGNAL;
   -- END;
END
|

DELIMITER ;
```

```
DELIMITER |

DROP TRIGGER IF EXISTS tr_policy_insert_end_date |
CREATE TRIGGER tr_policy_insert_end_date BEFORE
        INSERT ON ab_policy
   FOR EACH ROW
BEGIN
        DECLARE d DATETIME;
   SELECT
                a.start_date
        INTO d
   FROM
                ab_policy a
        WHERE
                a.policy_id = NEW.policy_id;

   IF (d IS NOT NULL AND d > NEW.end_date) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: The end date cannot be before the
start date';
        END IF;
END
|

DROP TRIGGER IF EXISTS tr_policy_update_end_date |
CREATE TRIGGER tr_policy_update_end_date BEFORE
        UPDATE ON ab_policy
   FOR EACH ROW
BEGIN
        DECLARE d DATETIME;
   SELECT
                a.start_date
        INTO d
   FROM
                ab_policy a
        WHERE
                a.policy_id = NEW.policy_id;

   IF (d IS NOT NULL AND d > NEW.end_date) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: The end date cannot be before the
start date';
        END IF;
```

```
END
|

DROP TRIGGER IF EXISTS tr_policy_insert_start_date |
CREATE TRIGGER tr_policy_insert_start_date BEFORE
        INSERT ON ab_policy
    FOR EACH ROW
BEGIN
        DECLARE d DATETIME;
    SELECT
                a.end_date
        INTO d
    FROM
                ab_policy a
        WHERE
                a.policy_id = NEW.policy_id;

        IF (d IS NOT NULL AND d < NEW.start_date) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: The Start date cannot be after the
end date';
        END IF;
END
|

DROP TRIGGER IF EXISTS tr_policy_update_start_date |
CREATE TRIGGER tr_policy_update_start_date BEFORE
        UPDATE ON ab_policy
    FOR EACH ROW
BEGIN
        DECLARE d DATETIME;
    SELECT
                a.end_date
        INTO d
    FROM
                ab_policy a
        WHERE
                a.policy_id = NEW.policy_id;

        IF (d IS NOT NULL AND d < NEW.start_date) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: The Start date cannot be after the
end date';
        END IF;
```

```
END
|

-- trigger to insure that AB_POLICY.PREMIUM is not negative

DROP TRIGGER IF EXISTS tr_policy_ins_prem |
CREATE TRIGGER tr_policy_ins_prem BEFORE
        INSERT ON ab_policy
   FOR EACH ROW
BEGIN
        IF (NEW.premium IS NULL OR NEW.premium < 0) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: Cannot create a policy with a
negative premium';
        END IF;
END
 |

DROP TRIGGER IF EXISTS tr_policy_upd_prem |
CREATE TRIGGER tr_policy_upd_prem BEFORE
        UPDATE ON ab_policy
   FOR EACH ROW
BEGIN
        IF (NEW.premium IS NULL OR NEW.premium <= 0) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: Cannot create a policy with a
negative premium';
        END IF;
END
 |

-- trigger to insure that AB_PAYMENT.AMOUNT is not negative

DROP TRIGGER IF EXISTS tr_pay_ins_amnt |
CREATE TRIGGER tr_pay_ins_amnt BEFORE
        INSERT ON ab_payment
   FOR EACH ROW
BEGIN
        IF (NEW.amount IS NULL OR NEW.amount <= 0) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'DB Error: Cannot create a payment that is 0 or
negative';
        END IF;
END
```

```
|

-- Trigger to deduct a payment from the total owed on an invoice

DROP TRIGGER IF EXISTS tr_pay_ins_updtotal |
CREATE TRIGGER tr_pay_ins_updtotal BEFORE
        INSERT ON ab_payment
   FOR EACH ROW FOLLOWS tr_pay_ins_amnt
BEGIN
        UPDATE ab_invoice
   SET total_paid = total_paid + NEW.amount
   WHERE invoice_id = NEW.invoice_id;
END

|

-- Trigger to deactivate an invoice if it has been paid off fully

DROP TRIGGER IF EXISTS tr_invoice_upd_deactivate |
CREATE TRIGGER tr_invoice_upd_deactivate BEFORE INSERT ON ab_invoice
FOR EACH ROW
BEGIN
        DECLARE d DECIMAL(9,2);
   DECLARE a DECIMAL(7,2);
        SELECT
                a.total_paid, a.amount
        INTO d, a
        FROM
                ab_invoice a
        WHERE
                a.invoice_id = NEW.invoice_id;

        IF (NEW.active <> 0 and d IS NOT NULL AND d <> NEW.total_paid AND
NEW.total_paid >= a) THEN
                SET NEW.active = 0;
        END IF;
END

|


DELIMITER ;
```

DML Script

The following DML script is used to populate seed data in the database.

```
USE ab_project;

INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(1, 'Alex', NULL, 'Biehl', 'M', 'M', 'H', '123 Some Street', NULL, 'Anytown', 'CA', '00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(2, 'Nick', 'E', 'Biehl', 'M', 'S', 'AH', '123 Some Street', 'APT B', 'Anytown', 'CA', '00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(3, 'Joe', NULL, 'Schmo', 'M', 'M', 'H', '123 Some Street', NULL, 'Anytown', 'CA', '00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(4, 'Tom', NULL, 'Bob', 'M', 'M', 'H', '123 Some Street', NULL, 'Anytown', 'CA', '00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(5, 'Richard', NULL, 'Long', 'M', 'M', 'H', '129 Some Street', NULL, 'Anytown', 'CA',
'00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(6, 'Harry', NULL, 'Mo', 'M', 'M', 'H', '123 Some Street', 'APT C', 'Anytown', 'CA', '00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(7, 'Abel', NULL, 'Tutor', 'M', 'M', 'H', '123 Some Street', NULL, 'Anytown', 'CA', '00000');
```

```
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(8, 'Abraham', NULL, 'Lincoln', 'M', 'M', 'H', '123 Some Street', NULL, 'Anytown', 'CA',
'00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(9, 'Melissa', 'R', 'Pena', 'F', 'M', 'H', '123 Some Street', NULL, 'Anytown', 'CA', '00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(10, 'Mathew', NULL, 'Taylor', 'M', 'M', 'H', '123 Some Street', NULL, 'Anytown', 'CA',
'00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(11, 'Chris', NULL, 'Harris', 'M', 'M', 'H', '123 Some Street', 'APT D', 'Anytown', 'CA',
'00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(12, 'Nirali', NULL, 'Patel', 'F', 'M', 'H', '123 Some Street', NULL, 'Anytown', 'CA', '00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(13, 'Madonna', NULL, 'Cosby', 'F', 'W', 'H', '123 Some Street', NULL, 'Anytown', 'CA',
'00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
	(14, 'Neva', NULL, 'Marsell', 'F', 'M', 'H', '123 Some Street', NULL, 'Anytown', 'CA',
'00000');
INSERT INTO ab_customer
	(cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
```

```
        (15, 'Tia', NULL, 'Lino', 'F', 'M', 'H', '223 Some Street', NULL, 'Anytown', 'CA', '00000');
INSERT INTO ab_customer
        (cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
        (16, 'Gayla', NULL, 'Gaimer', 'F', 'M', 'H', '123 Some Street', NULL, 'Anytown', 'CA',
'00000');
INSERT INTO ab_customer
        (cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
        (17, 'Trey', NULL, 'Trout', 'M', 'M', 'H', '153 Some Street', NULL, 'Anytown', 'CA', '00000');
INSERT INTO ab_customer
        (cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
        (18, 'Rosemarie', NULL, 'Fields', 'F', 'M', 'AH', '123 Some Street', NULL, 'Anytown', 'CA',
'00000');
INSERT INTO ab_customer
        (cust_id, fname, mname, lname, gender, marital_status, cust_type, street_1, street_2,
city, state, zip)
VALUES
        (19, 'Teri', NULL, 'Erlwile', 'F', 'S', 'AH', '126Some Street', NULL, 'Anytown', 'CA', '00000');

COMMIT;

-- ab policy inserts

INSERT INTO ab_policy
        (policy_id, p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
        (1, 'AH', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(),
INTERVAL 0 YEAR), 1000.00, 'C', 1, 1);
INSERT INTO ab_policy
        (policy_id, p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
        (2, 'AH', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(),
INTERVAL 0 YEAR), 1000.00, 'C', 1, 2);
INSERT INTO ab_policy
        (policy_id, p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
        (3, 'AH', ADDDATE(SYSDATE(), INTERVAL -5 YEAR), ADDDATE(SYSDATE(),
INTERVAL -0 YEAR), 1000.00, 'C', 1, 3);
INSERT INTO ab_policy
```

```
	(policy_id, p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	(4, 'A', ADDDATE(SYSDATE(), INTERVAL -2 YEAR), ADDDATE(SYSDATE(),
INTERVAL -1 YEAR), 1000.00, 'P', 0, 4);
INSERT INTO ab_policy
	(policy_id, p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	(5, 'AH', ADDDATE(SYSDATE(), INTERVAL -10 YEAR), ADDDATE(SYSDATE(),
INTERVAL -0 YEAR), 1000.00, 'C', 1, 4);
INSERT INTO ab_policy
	(policy_id, p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	(6, 'AH', ADDDATE(SYSDATE(), INTERVAL -10 YEAR), ADDDATE(SYSDATE(),
INTERVAL -5 YEAR), 1000.00, 'P', 0, 5);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'H', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(), INTERVAL
-0 YEAR), 1000.00, 'C', 1, 5);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'A', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(), INTERVAL
-0 YEAR), 1000.00, 'C', 1, 6);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'AH', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(), INTERVAL
-0 YEAR), 1000.00, 'C', 1, 7);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'H', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(), INTERVAL
-0 YEAR), 1000.00, 'C', 1, 8);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'A', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(), INTERVAL
-0 YEAR), 1000.00, 'C', 1, 9);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'AH', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(), INTERVAL
-0 YEAR), 1000.00, 'C', 1, 10);
```

```
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'H', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(),  INTERVAL
-0 YEAR), 1000.00, 'C', 1, 11);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'AH', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(),  INTERVAL
-0 YEAR), 1000.00, 'C', 1, 12);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'AH', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(),  INTERVAL
-0 YEAR), 1000.00, 'C', 1, 13);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'AH', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(),  INTERVAL
-0 YEAR), 1000.00, 'C', 1, 14);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'AH', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(),  INTERVAL
-0 YEAR), 1000.00, 'C', 1, 15);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'AH', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(),  INTERVAL
-0 YEAR), 1000.00, 'C', 1, 16);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'A', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(),  INTERVAL
-0 YEAR), 1000.00, 'C', 1, 1);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
	( 'AH', ADDDATE(SYSDATE(), INTERVAL -1 YEAR), ADDDATE(SYSDATE(),  INTERVAL
-0 YEAR), 1000.00, 'C', 1, 17);
INSERT INTO ab_policy
	(p_type, start_date, end_date, premium, state, active, cust_id)
VALUES
```

```
        ( 'AH', ADDDATE(SYSDATE(), INTERVAL -7 YEAR), ADDDATE(SYSDATE(), INTERVAL
-2 YEAR), 1000.00, 'P', 0, 18);

COMMIT;

-- insert values into ab_home and ab_auto for their respective policies

INSERT INTO ab_home (policy_id) VALUES (1);
INSERT INTO ab_home (policy_id) VALUES (2);
INSERT INTO ab_home (policy_id) VALUES (3);
INSERT INTO ab_home (policy_id) VALUES (5);
INSERT INTO ab_home (policy_id) VALUES (6);
INSERT INTO ab_home (policy_id) VALUES (7);
INSERT INTO ab_home (policy_id) VALUES (9);
INSERT INTO ab_home (policy_id) VALUES (10);
INSERT INTO ab_home (policy_id) VALUES (12);
INSERT INTO ab_home (policy_id) VALUES (13);
INSERT INTO ab_home (policy_id) VALUES (14);
INSERT INTO ab_home (policy_id) VALUES (15);
INSERT INTO ab_home (policy_id) VALUES (16);
INSERT INTO ab_home (policy_id) VALUES (17);
INSERT INTO ab_home (policy_id) VALUES (18);
INSERT INTO ab_home (policy_id) VALUES (20);
INSERT INTO ab_home (policy_id) VALUES (21);

INSERT INTO ab_auto (policy_id) VALUES (1);
INSERT INTO ab_auto (policy_id) VALUES (2);
INSERT INTO ab_auto (policy_id) VALUES (3);
INSERT INTO ab_auto (policy_id) VALUES (4);
INSERT INTO ab_auto (policy_id) VALUES (5);
INSERT INTO ab_auto (policy_id) VALUES (6);
INSERT INTO ab_auto (policy_id) VALUES (8);
INSERT INTO ab_auto (policy_id) VALUES (9);
INSERT INTO ab_auto (policy_id) VALUES (11);
INSERT INTO ab_auto (policy_id) VALUES (12);
INSERT INTO ab_auto (policy_id) VALUES (14);
INSERT INTO ab_auto (policy_id) VALUES (15);
INSERT INTO ab_auto (policy_id) VALUES (16);
INSERT INTO ab_auto (policy_id) VALUES (17);
INSERT INTO ab_auto (policy_id) VALUES (18);
INSERT INTO ab_auto (policy_id) VALUES (19);
INSERT INTO ab_auto (policy_id) VALUES (20);
INSERT INTO ab_auto (policy_id) VALUES (21);
```

```
COMMIT;

-- INSERT INTO AB_HOUSE

INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -1 YEAR), 1000000.00, 2000, 'S', 1, 1, NULL, 0, 1);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -2 YEAR), 2500000.00, 2000, 'M', 1, 1, 'O', 0, 2);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -7 YEAR), 1500000.00, 2000, 'C', 1, 1, NULL, 0, 3);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -3 YEAR), 1000000.00, 2000, 'T', 1, 1, 'U', 0, 5);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -9 YEAR), 1000000.00, 2000, 'S', 1, 1, NULL, 0, 6);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -20 YEAR), 1000000.00, 2000, 'M', 1, 1, NULL, 0,
7);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -40 YEAR), 1000000.00, 2000, 'S', 1, 1, NULL, 0,
9);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
```

```
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -15 YEAR), 1000000.00, 2000, 'S', 1, 1, NULL, 0,
10);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -20 YEAR), 100000.00, 2000, 'S', 1, 1, NULL, 0,
12);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -3 YEAR), 1520000.00, 2000, 'S', 1, 1, NULL, 0,
13);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -8 YEAR), 5000000.00, 2000, 'S', 1, 1, 'I', 0, 14);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -2 YEAR), 1000000.00, 2000, 'S', 1, 1, 'O', 0, 15);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -1 YEAR), 250000.00, 2000, 'S', 1, 1, 'M', 0, 16);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -10 YEAR), 1000000.00, 2000, 'S', 1, 1, NULL, 0,
17);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -13 YEAR), 500000.00, 2000, 'S', 1, 1, 'U', 0, 18);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
```

```
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -1 YEAR), 2000000.00, 2000, 'S', 1, 1, 'M', 0, 20);
INSERT INTO ab_house
	(purchase_date, purchase_value, area, house_type, auto_fire_notif, home_security,
pool, basement, policy_id)
VALUES
	(ADDDATE(SYSDATE(), INTERVAL -1 YEAR), 1000000.00, 2000, 'S', 1, 1, 'O', 0, 21);

COMMIT;


-- Create AB_Driver records
INSERT INTO ab_driver
	(license, fname, mname, lname, birthdate)
VALUES
	( 'K3L 4B9', 'Raphael', NULL, 'Owens', '2021-02-14 15:36:27');
INSERT INTO ab_driver
	(license, fname, mname, lname, birthdate)
VALUES
	( 'P4U 5B4', 'Berk', 'D', 'Meadows', '2021-06-25 17:24:48');
INSERT INTO ab_driver
	(license, fname, mname, lname, birthdate)
VALUES
	( 'R2B 3A4', 'Jonah', NULL, 'Potts', '2020-08-29 20:25:25');
INSERT INTO ab_driver
	(license, fname, mname, lname, birthdate)
VALUES
	( 'X9B 0H6', 'Patience', NULL, 'Nelson', '2020-09-10 02:06:27');
INSERT INTO ab_driver
	(license, fname, mname, lname, birthdate)
VALUES
	( 'S0I 2S3', 'Morgan', 'B', 'Brennan', '2020-09-04 11:29:41');
INSERT INTO ab_driver
	(license, fname, mname, lname, birthdate)
VALUES
	( 'A9M 4Q6', 'Kristen', NULL, 'Dudley', '1990-11-03 23:32:21');
INSERT INTO ab_driver
	(license, fname, mname, lname, birthdate)
VALUES
	( 'Z2D 4D3', 'Randall', 'R', 'Mcbride', '2000-09-14 05:25:04');
INSERT INTO ab_driver
	(license, fname, mname, lname, birthdate)
VALUES
	( 'Z2X 2Y1', 'Merrill', NULL, 'Waters', '2001-05-06 04:01:02');
```

```
INSERT INTO ab_driver
	(license, fname, mname, lname, birthdate)
VALUES
	( 'T0W 3X6', 'Alec', NULL, 'Estrada', '1980-08-30 14:37:30');
INSERT INTO ab_driver
	(license, fname, mname, lname, birthdate)
VALUES
	( 'F1U 5Y6', 'Steve', 'M', 'Guy', '1985-09-20 00:00:00');
INSERT INTO ab_driver
	(license, fname, mname, lname, birthdate)
VALUES
	( 'F1U 5Y7', 'Lucy', 'F', 'Chick', '2000-09-09 06:30:00');

COMMIT;

-- Create AB_Vehicle records

INSERT INTO ab_vehicle
	(vin, make, model, year, state, policy_id)
VALUES
	( '628047588812090', 'Honda', 'Civic', 2012, 'O', 1);
INSERT INTO ab_vehicle
	(vin, make, model, year, state, policy_id)
VALUES
	( '232743502846810', 'Honda', 'Accord', 2019, 'L', 2);
INSERT INTO ab_vehicle
	(vin, make, model, year, state, policy_id)
VALUES
	( '777460265379710', 'Tesla', 'Model 3', 2000, 'F', 3);
INSERT INTO ab_vehicle
	(vin, make, model, year, state, policy_id)
VALUES
	( '013385760649101', 'Tesla', 'Model S', 1990, 'O', 4);
INSERT INTO ab_vehicle
	(vin, make, model, year, state, policy_id)
VALUES
	( '654499508054630', 'Audi', 'A8', 1995, 'L', 5);
INSERT INTO ab_vehicle
	(vin, make, model, year, state, policy_id)
VALUES
	( '496486322739760', 'Audi', 'A6', 2000, 'L', 6);
INSERT INTO ab_vehicle
	(vin, make, model, year, state, policy_id)
VALUES
```

```
        ( '298326397808910', 'Mazda', 'Miata', 1980, 'O', 8);

-- cfeate AB_DRIVER_VEHILCE records

INSERT INTO ab_driver_vehicle
        (license, vin)
VALUES
        ( 'K3L 4B9', '628047588812090');
INSERT INTO ab_driver_vehicle
        (license, vin)
VALUES
        (  'P4U 5B4', '232743502846810');
INSERT INTO ab_driver_vehicle
        (license, vin)
VALUES
        ( 'R2B 3A4', '777460265379710');
INSERT INTO ab_driver_vehicle
        (license, vin)
VALUES
        ( 'X9B 0H6', '013385760649101');
INSERT INTO ab_driver_vehicle
        (license, vin)
VALUES
        ( 'T0W 3X6', '654499508054630');
INSERT INTO ab_driver_vehicle
        (license, vin)
VALUES
        ( 'F1U 5Y6', '496486322739760');
INSERT INTO ab_driver_vehicle
        (license, vin)
VALUES
        ( 'F1U 5Y7', '298326397808910');

-- CREATE AB_INVOICE RECORDS

INSERT INTO ab_invoice
        (invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
        (6341841,"2021-05-11 03:28:43","6466.74","2021-03-25 09:34:57","738.35",0,8);
INSERT INTO
        ab_invoice
(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
        (6375949,"2020-09-04 00:35:36","2089.72","2020-09-11 20:21:11","298.77",1,12);
```

```
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(435513,"2020-11-19 23:22:12","5572.22","2020-09-13 08:40:27","998.58",0,10);
INSERT INTO
	ab_invoice
(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(4821965,"2020-12-13 09:58:36","5247.20","2020-09-13 15:13:12","156.12",1,15);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(8798943,"2021-07-19 20:37:17","5491.30","2021-07-18 23:33:13","891.48",0,19);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(879003,"2020-11-29 01:43:54","1155.29","2021-03-29 21:46:12","631.54",1,12);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(3984561,"2020-09-02 14:45:43","6974.45","2021-02-10 00:50:43","769.10",0,6);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(2932905,"2020-08-06 06:03:11","3130.73","2020-10-16 19:24:14","117.46",0,12);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(5918869,"2021-04-01 00:39:51","2229.73","2021-07-05 08:34:25","785.56",1,17);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(8045261,"2021-02-13 11:17:39","3539.08","2021-06-28 03:40:31","469.49",0,16);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(7308349,"2020-10-18 03:03:01","1333.44","2020-11-01 20:17:20","739.16",0,7);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(6238400,"2020-08-24 16:52:28","1713.31","2020-08-25 22:11:33","119.99",1,9);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
```

```
	(183347,"2021-02-07 00:58:41","4129.73","2020-09-12 04:22:52","836.25",0,15);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(3132230,"2021-04-28 00:49:58","3361.45","2021-01-28 19:56:20","283.68",1,6);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(2515970,"2020-09-20 05:49:50","2691.34","2021-05-16 21:22:17","962.56",1,13);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(1071562,"2021-05-08 22:50:11","9202.28","2021-05-16 08:41:44","170.70",1,5);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(5921532,"2021-04-21 01:31:14","6068.50","2021-04-26 04:20:07","987.32",1,15);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(2795747,"2021-03-18 14:45:16","7546.56","2020-08-25 16:23:48","881.80",0,2);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(9273101,"2021-05-08 11:06:11","6820.46","2021-05-30 09:49:41","471.18",1,1);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(1846222,"2020-07-30 16:32:45","6035.68","2020-10-03 16:46:53","692.43",1,1);
INSERT INTO ab_invoice
	(invoice_id,invoice_date,amount,payment_date,total_paid,active,policy_id)
VALUES
	(5877124,"2021-05-20 21:55:13","9118.48","2021-02-25 08:36:53","622.81",0,13);

-- CREATE AB_PAYMENT RECORDS

INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-05-27 06:40:24","7866.05","PayPal",6341841);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-01-28 22:34:18","8130.53","PayPal",5921532);
```

```
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-03-20 22:18:33","4130.77","Debit",2795747);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-04-24 03:33:12","1535.49","PayPal",1846222);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2020-10-25 04:58:00","6570.28","Check",1846222);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2020-12-28 19:12:03","4258.54","Credit",5877124);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-02-01 07:28:50","4473.05","Check",3132230);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2020-10-26 13:07:34","936.56","PayPal",6238400);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2020-12-13 23:20:22","7760.62","PayPal",6375949);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-03-10 04:36:05","4781.91","PayPal",435513);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-07-15 18:53:46","1703.17","Credit",4821965);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-01-28 09:49:01","8697.58","Credit",8798943);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2020-08-18 11:23:53","7820.75","Debit",879003);
```

```
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-07-14 03:25:05","407.92","Check",3984561);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2020-08-02 21:45:43","3507.55","PayPal",2932905);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-07-05 05:55:37","361.29","Debit",5918869);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-06-15 13:17:07","2754.62","PayPal",8045261);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2020-08-29 06:55:23","9276.79","Check",7308349);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-02-11 21:49:15","1289.55","PayPal",6238400);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2021-07-25 13:13:18","5385.38","Debit",183347);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2020-08-04 14:09:23","5945.09","Check",3132230);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES
	("2020-08-26 04:37:02","8912.51","Credit",2515970);
INSERT INTO ab_payment
	(pay_date,amount,pay_type,invoice_id)
VALUES ("2021-02-06 02:52:41","7339.07","Credit",1071562);

COMMIT;
```

Record Counts

AB_CUSTOMER

SELECT COUNT(*) FROM ab_customer;

COUNT(*)
19

AB_POLICY

SELECT COUNT(*) FROM ab_policy;

COUNT(*)
21

AB_HOME

SELECT COUNT(*) FROM ab_home;

COUNT(*)
17

AB_AUTO

SELECT COUNT(*) FROM ab_auto;

COUNT(*)
18

AB_HOUSE

SELECT COUNT(*) FROM AB_HOUSE;

COUNT(*)
18

AB_INVOICE

SELECT COUNT(*) FROM ab_invoice;

COUNT(*)
21

AB_PAYMENT

SELECT COUNT(*) FROM ab_payment;

COUNT(*)
23

AB_VEHICLE

SELECT COUNT(*) FROM ab_vehicle;

COUNT(*)
7

AB_DRIVER

SELECT COUNT(*) FROM ab_driver;

COUNT(*)
11

AB_DRIVER_VEHICLE

SELECT COUNT(*) FROM ab_driver_vehicle;

COUNT(*)
7

Data Dictionary Queries

Table Dictionary

```
SELECT
	table_name, table_rows, avg_row_length, auto_increment
FROM
	INFORMATION_SCHEMA.TABLES
WHERE
	table_schema LIKE 'ab_project'
ORDER BY
	table_name;
```

table_name	**table_rows**	**avg_row_length**	**auto_increment**
ab_auto	18	910	22
ab_customer	19	862	20
ab_driver	11	1489	NULL
ab_driver_vehicle	7	2340	NULL

ab_home	17	963	NULL
ab_house	17	964	18
ab_invoice	21	780	9273102
ab_payment	23	712	24
ab_policy	21	780	22
ab_vehicle	7	2340	NULL

Column Dictionary

```
SELECT
	table_name, column_name, column_default, is_nullable, data_type,  column_type, column_comment
FROM
	INFORMATION_SCHEMA.COLUMNS
WHERE
	table_name LIKE 'ab_%'
ORDER BY
	table_name, column_name;
```

table_name	column_name	column_default	is_nullable	data_type	column_type	column_comment
ab_auto	policy_id	NULL	NO	int	int(11)	
ab_customer	city	NULL	NO	varchar	varchar(32)	THE CUSTOMERS ADDRESS CITY.
ab_customer	cust_id	NULL	NO	int	int(11)	THE CUSTOMERS UNIQUE ID
ab_customer	cust_type	NULL	NO	varchar	varchar(2)	THE CUSTOMER TYPE. 'A' AUTOMOBILE INSURANCE,

						'H' HOME INSURANCE.
ab_customer	fname	NULL	NO	varchar	varchar(32)	THE CUSTOMERS FIRST NAME
ab_customer	gender	NULL	YES	varchar	varchar(1)	THE CUSTOMERS GENDER
ab_customer	lname	NULL	NO	varchar	varchar(32)	THE CUSTOMERS LAST NAME
ab_customer	marital_status	NULL	NO	varchar	varchar(1)	THE CUSTOMERS MARITAL STATUS. EITHER 'M', 'S', OR 'W'
ab_customer	mname	NULL	YES	varchar	varchar(1)	THE CUSTOMERS MIDDLE INITIAL
ab_customer	state	NULL	NO	varchar	varchar(2)	THE CUSTOMERS HOME ADDRESS STATE CODE
ab_customer	street_1	NULL	NO	varchar	varchar(32)	CUSTOMERS STREET ADDRESS
ab_customer	street_2	NULL	YES	varchar	varchar(32)	OPTIONAL CUSTOMER FLOOR/APARTMENT.
ab_customer	zip	NULL	NO	varchar	varchar(5)	THE CUSTOMERS HOME ADDRESS ZIPCODE.
ab_driver	birthdate	NULL	NO	datetime	datetime	THE DRIVERS BIRTHDATE.
ab_driver	fname	NULL	NO	varchar	varchar(32)	THE DRIVERS

						FIRST NAME.
ab_driver	license	NULL	NO	varchar	varchar(16)	THE DRIVERS LICENSE
ab_driver	lname	NULL	NO	varchar	varchar(32)	THE DRIVERS LAST NAME.
ab_driver	mname	NULL	YES	varchar	varchar(1)	THE DRIVERS OPTIONAL MIDDLE INITIAL.
ab_driver_vehicle	license	NULL	NO	varchar	varchar(16)	THE LICENSE OF THE CARS DRIVER
ab_driver_vehicle	vin	NULL	NO	varchar	varchar(17)	THE VIN OF THE INSURED VEHICLE
ab_home	policy_id	NULL	NO	int	int(11)	THE INSURANCE POLICY UNIQUE ID
ab_house	area	NULL	NO	decimal	decimal(7,2)	THE HOMES AREA IN SQUARE FEET.
ab_house	auto_fire_notif	NULL	NO	tinyint	tinyint(4)	WHETHER THE HOUSE HAS AUTOMATIC FIRE NOTIFICATION TO THE FIRE DEPARTMENT.
ab_house	basement	NULL	NO	tinyint	tinyint(4)	WHETHER THE HOUSE HAS A BASEMENT.
ab_house	home_id	NULL	NO	int	int(11)	THE HOUSES UNIQUE ID
ab_house	home_security	NULL	NO	tinyint	tinyint(4)	WHETHER THE HOUSE

						HAS A SECURITY SYSTEM.
ab_house	house_type	NULL	NO	varchar	varchar(1)	THE HOME TYPE. 'S' IS SINGLE FAMILY, 'M' IS MULTI FAMILY, 'C' IS CONDOMINIUM, 'T' IS TOWN HOUSE.
ab_house	policy_id	NULL	NO	int	int(11)	THE ID OF THE POLICY INSURING THE HOUSE
ab_house	pool	NULL	YES	varchar	varchar(1)	SWIMMING POOL. 'U' IS UNDERGROUND, 'O' IS OVERGROUND, 'I' IS INDOOR, 'M' IS MULTIPLE, NULL IS NO POOL.
ab_house	purchase_date	NULL	NO	datetime	datetime	THE DATE THE HOME WAS PURCHASED.
ab_house	purchase_value	NULL	NO	decimal	decimal(9,2)	THE HOMES PURCHASE VALUE.
ab_invoice	active	NULL	NO	tinyint	tinyint(4)	WHETHER THE INVOICE IS ACTIVE
ab_invoice	amount	NULL	NO	decimal	decimal(7,2)	THE AMOUNT DUE
ab_invoice	invoice_date	NULL	NO	datetime	datetime	THE DATE GENERATED.
ab_invoice	invoice_id	NULL	NO	int	int(11)	THE INVOICES

						UNIQUE ID
ab_invoice	payment_date	NULL	NO	datetime	datetime	THE DATE THE INVOICE IS DUE.
ab_invoice	policy_id	NULL	NO	int	int(11)	ID OF THE POLICY THAT THE INVOICE BELONGS TO
ab_invoice	total_paid	NULL	NO	decimal	decimal(9,2)	The amount that the client has paid so far.
ab_payment	amount	NULL	NO	decimal	decimal(7,2)	THE PAYMENT INSTALLMENT AMOUNT.
ab_payment	invoice_id	NULL	NO	int	int(11)	THE INVOICE THE PAYMENT IS GOING TOWARDS
ab_payment	pay_date	NULL	NO	datetime	datetime	THE DATE THE PAYMENT WAS MADE
ab_payment	pay_type	NULL	NO	varchar	varchar(6)	THE METHOD OF PAYMENT; ONE OF 'PayPal', 'Credit', 'Debit', OR 'Check'.
ab_payment	p_id	NULL	NO	bigint	bigint(20)	THE PAYMENTS UNIQUE ID
ab_policy	active	NULL	NO	tinyint	tinyint(4)	WHETHER THE POLICY IS STILL ACTIVE
ab_policy	cust_id	NULL	NO	int	int(11)	ID OF THE CUSTOMER HOLDING THE POLICY

ab_policy	end_date	NULL	NO	datetime	datetime	THE POLICY END DATE.
ab_policy	policy_id	NULL	NO	int	int(11)	THE POLICYS UNIQUE ID
ab_policy	premium	NULL	NO	decimal	decimal(7,2)	THE PREMIUM AMOUNT.
ab_policy	p_type	NULL	NO	varchar	varchar(9)	THE POLICY TYPE. 'A' FOR AUTO AND 'H' FOR HOME.
ab_policy	start_date	NULL	NO	datetime	datetime	THE POLICY START DATE
ab_policy	state	NULL	NO	varchar	varchar(1)	THE POLICY STATUS. 'C' FOR CURRENT, 'P' FOR EXPIRED.
ab_vehicle	make	NULL	NO	varchar	varchar(32)	THE VEHICLE MAKE.
ab_vehicle	model	NULL	NO	varchar	varchar(32)	THE VEHICLE MODEL.
ab_vehicle	policy_id	NULL	NO	int	int(11)	THE ID OF THE POLICY INSURING THE CAR
ab_vehicle	state	NULL	NO	varchar	varchar(1)	VEHICLE STATUS. 'L' IS LEASED, 'F' IS FINANCED, AND 'O' IS OWNED.
ab_vehicle	vin	NULL	NO	varchar	varchar(17)	THE UNIQUE VEHICLE IDENTIFICATIO N NUMBER
ab_vehicle	year	NULL	NO	smallint	smallint(6)	THE VEHICLE YEAR.

Constraint Dictionary

```
SELECT *
FROM
	INFORMATION_SCHEMA.TABLE_CONSTRAINTS
WHERE
	table_name LIKE 'ab_%'
ORDER BY
	table_name, constraint_name;
```

CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE
def	ab_project	ab_auto_ab_policy_fk	ab_project	ab_auto	FOREIGN KEY
def	ab_project	policy_id	ab_project	ab_auto	UNIQUE
def	ab_project	PRIMARY	ab_project	ab_auto	PRIMARY KEY
def	ab_project	cust_id	ab_project	ab_customer	UNIQUE
def	ab_project	PRIMARY	ab_project	ab_customer	PRIMARY KEY
def	ab_project	license	ab_project	ab_driver	UNIQUE
def	ab_project	PRIMARY	ab_project	ab_driver	PRIMARY KEY
def	ab_project	ab_driver_ab_vehicle_fk	ab_project	ab_driver_vehicle	FOREIGN KEY
def	ab_project	ab_driver_vehicle_ab_driver_fk	ab_project	ab_driver_vehicle	FOREIGN KEY
def	ab_project	PRIMARY	ab_project	ab_driver_vehicle	PRIMARY KEY
def	ab_project	ab_home_ab_polic	ab_project	ab_home	FOREIGN

		y_fk			KEY
def	ab_project	PRIMARY	ab_project	ab_home	PRIMARY KEY
def	ab_project	ab_house_ab_home_fk	ab_project	ab_house	FOREIGN KEY
def	ab_project	home_id	ab_project	ab_house	UNIQUE
def	ab_project	PRIMARY	ab_project	ab_house	PRIMARY KEY
def	ab_project	ab_invoice_ab_policy_fk	ab_project	ab_invoice	FOREIGN KEY
def	ab_project	invoice_id	ab_project	ab_invoice	UNIQUE
def	ab_project	PRIMARY	ab_project	ab_invoice	PRIMARY KEY
def	ab_project	ab_payment_ab_invoice_fk	ab_project	ab_payment	FOREIGN KEY
def	ab_project	PRIMARY	ab_project	ab_payment	PRIMARY KEY
def	ab_project	p_id	ab_project	ab_payment	UNIQUE
def	ab_project	ab_policy_ab_customer_fk	ab_project	ab_policy	FOREIGN KEY
def	ab_project	policy_id	ab_project	ab_policy	UNIQUE
def	ab_project	PRIMARY	ab_project	ab_policy	PRIMARY KEY
def	ab_project	ab_vehicle_ab_auto_fk	ab_project	ab_vehicle	FOREIGN KEY
def	ab_project	PRIMARY	ab_project	ab_vehicle	PRIMARY KEY
def	ab_project	vin	ab_project	ab_vehicle	UNIQUE

Summary

This design encapsulates the database model that will be used to bring We Do Secure's (WDS) business growth to life. Each customer, stored in the AB_CUSTOMER table, will have one or more insurance policies associated with them. Policies can be either Home or Auto policies, based on the type. Since all of the attributes in a home or auto policy are the same, we made the supertype entity the AB_POLICY entity, and the extending subtypes being AB_HOME or AB_AUTO. Logic has been put in place so that a House record in the AB_HOUSE table cannot be associated with a Policy that is not a Housing policy type, and likewise for records in AB_VEHICLE, but with the restriction being that the associated policy is an Auto policy. Since each vehicle may have multiple drivers, we stored driver information in the AB_DRIVER table, which an intersect entity, AB_DRIVER_VEHICLE, used to encapsulate the many-to-many relationship between Drivers and Vehicles. Lastly, all invoices generated for Insurance Policies are stored in the AB_INVOICE table, with a foreign key reference to the policy that they belong to. Each Invoice record will have one or more Payments associated with it, since a customer is allowed to pay each invoice either in a single payment, or in multiple. Each payment will be stored in the AB_PAYMENT table, with a foreign key reference to the Invoice that that particular payment was for. Business rules have been put in place on the AB_PAYMENT table, such that when a payment is submitted, the amount paid is automatically added to the Total Paid amount on the invoice. Additionally, when the Total Paid amount on an Invoice matches the amount due, the invoice is considered paid off, and it is automatically deactivated by setting the Invoice record's Active flag to false. Constraints have also been put on the AB_PAYMENT table such that the amount attribute cannot be zero or less than zero. Lastly, constraints have been put in place on the AB_POLICY table to ensure that the Start Date cannot be after the End Date, and vice versa.