Разработка триггеров:

1) Расчет числа на +/- в текущей таблице:

- При вставке нового заказа, проверяется наличие других заказов у данного заказчика, и при наличии некоторого количества заказов (берётся из другой таблицы), делается скидка при производстве заказа (также берётся из другой таблицы).

2) Проверка даты в текущей таблице (дата должна рассчитываться):

- При вставке нового заказа, срок сдачи заказа формируется из максимального срока поставки + срок создания по техническому процессу. Если невозможно рассчитать, то срок NULL. Если реальный срок сдачи не соответствует предполагаемому сроку, то он меняется на предполагаемый. Если соответствует, то остаётся реальный срок.

3) Автоматическая вставка данных в связанную таблицу (не в текущую таблицу):

- При вставке нового заказа, в отдельной таблице гарантийных талонов создаётся новый гарантийный талон на случай, если заказ не будет выполнен в срок.

4) Автоматическое удаление данных из связанных/несвязанных таблиц (допускается каскадное удаление):

- При удалении заказчика, все его заказы аннулируются, и тем самым удаляются из таблицы заказов с проверкой.

5) Формирование строки из данных других строк в текущей таблице:

- Формирование предзаказа. При создании нового заказа, он сначала добавляется в таблицу предзаказов, а его номер формируется из номера заказ и номера заказчика.

6) Проверка на соответствие числового значения не в текущей таблице:

- При создании продукта, проверяется наличие необходимого количества материалов, и если их не хватает, выводится минимальная стоимость закупки у подрядчиков из отдельной таблицы.

7) Формирование даты из строковых данных этой же таблицы:

- Таблица сроков сдачи продуктов. Пользователь вводит ID продукта, его количество и день, месяц и год сроков сдачи этого количества продуктов. Из этих данных формируется дата сдачи.

Код триггеров:

1) Расчет числа на +/- в текущей таблице:

```
CREATE TABLE Discounts
(
    OrderAmount integer UNIQUE NOT NULL,
    Discount    integer        NOT NULL
);

INSERT INTO Discounts
VALUES (0, 500),
       (1, 1000);

CREATE TRIGGER DiscountOnOrder
    BEFORE INSERT
    on Orders
    FOR EACH ROW
BEGIN
    SET @AmountOfOrders = (SELECT COUNT(OrderID) FROM Orders WHERE CustomerID =
NEW.CustomerID);
    SET @Discount = (SELECT (Discount)
                     from Discounts
                     Where OrderAmount = (SELECT MAX(OrderAmount) FROM Discounts WHERE
OrderAmount <= @AmountOfOrders));
    SET NEW.OrderPrice = (NEW.OrderPrice - IFNULL(@Discount, 0));
end;
```

2) Проверка даты в текущей таблице (дата должна рассчитываться):

```
CREATE TABLE Technologies
(
    TechnologyID       integer UNIQUE AUTO_INCREMENT NOT NULL,
    ImplementationDays integer DEFAULT 0
);

INSERT INTO Technologies (ImplementationDays) VALUE (45);

ALTER TABLE Orders
    ADD FOREIGN KEY (TechnologyID) REFERENCES Technologies (TechnologyID);

CREATE TRIGGER OrderDate
    BEFORE INSERT
    on Orders
    FOR EACH ROW
begin
    set @AllegedDate = DATE_ADD(NEW.OrderStartDate, INTERVAL (SELECT (ImplementationDays)
                                                              FROM Technologies
                                                              WHERE NEW.TechnologyID =
Technologies.TechnologyID) DAY);
    set NEW.OrderEndDate = IF(NEW.OrderEndDate > @AllegedDate, @AllegedDate,
NEW.OrderEndDate);
end;
```

3) Автоматическая вставка данных в связанную таблицу (не в текущую таблицу):

```
CREATE TABLE ServiceTalons
(
    TalonID integer UNIQUE PRIMARY KEY AUTO_INCREMENT,
    OrderID integer UNIQUE
);

ALTER TABLE ServiceTalons
    ADD FOREIGN KEY (OrderID) REFERENCES Orders (OrderID);

CREATE TRIGGER OnOrderInsert_Talon
    AFTER INSERT
    on Orders
    FOR EACH ROW
begin
    INSERT INTO ServiceTalons (OrderID) VALUE (NEW.OrderID);
end;

CREATE TRIGGER OnOrderDelete_Talon
    BEFORE DELETE
    on Orders
    FOR EACH ROW
begin
    DELETE FROM ServiceTalons WHERE ServiceTalons.OrderID = OLD.OrderID;
end;
```

4) Автоматическое удаление данных из связанных/несвязанных таблиц (допускается каскадное удаление):

```
CREATE TRIGGER OnCustomerDelete
    BEFORE DELETE
    on Customers
    FOR EACH ROW
begin
    DELETE FROM Orders WHERE Orders.CustomerID = OLD.CustomerID;
end;
```

5) Формирование строки из данных других строк в текущей таблице:

```
CREATE TABLE PreOrders
(
    PreOrderID VARCHAR(64) NOT NULL UNIQUE PRIMARY KEY
);

CREATE TRIGGER OnOrderInsert_PREORDER
    AFTER INSERT
    on Orders
    FOR EACH ROW
begin
    INSERT INTO PreOrders VALUE (CONCAT(CAST(NEW.OrderID as CHAR), '_', CAST(NEW.CustomerID as
CHAR)));
end;

CREATE TRIGGER OnOrderDelete_PREORDER
    BEFORE DELETE
    on Orders
    FOR EACH ROW
begin
    DELETE
    FROM PreOrders
    WHERE PreOrderID =
          (CONCAT(CAST(OLD.OrderID as CHAR), '_', CAST(OLD.CustomerID as CHAR)));
end;
```

6) Проверка на соответствие числового значения не в текущей таблице:

```
CREATE TRIGGER Product2MaterialInsert
    BEFORE INSERT
    on Product2Material
    for each row
begin
    SET @MaterialAmount = (SELECT (MaterialAmount) FROM Materials WHERE MaterialID =
NEW.MaterialID);
    IF (NEW.MaterialQuantity > @MaterialAmount) THEN
        SET @MinPrice = (SELECT MIN(MaterialPrice) FROM SupplierMaterialPrice WHERE MaterialID
= NEW.MaterialID) * (NEW.MaterialQuantity - @MaterialAmount);
        SET @msgtext = CONCAT('For production, you would need to buy materials for ',
CAST(@MinPrice as CHAR));
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = @msgtext;
    end if;
end;
```

7) Формирование даты из строковых данных этой же таблицы:

```
Create TABLE OrderProductTime
(
    OrderProductID integer not null auto_increment primary key unique,
    ProductID      integer not null,
    ProductAmount  integer not null,
    DueDay         integer not null,
    DueMonth       integer not null,
    DueYear        integer not null,
    DueDate        date
);

ALTER TABLE OrderProductTime
    ADD FOREIGN KEY (ProductID) REFERENCES Products (ProductID);

CREATE TRIGGER OnOrderProduct
    BEFORE INSERT
    On OrderProductTime
    for each row
begin
    SET NEW.DueDate = STR_TO_DATE(CONCAT(NEW.DueYear, '-', NEW.DueMonth, '-', NEW.DueDay),
'%Y-%m-%d');
end;
```