Date	**20-06-2025**
Team ID	**LTVIP2025TMID29686**
Project Name	**Field Service WorkOrder Optimization**
Maximum Marks	

Chapter-12

Appendix (Apex code)

1. WorkOrderClass(apex)

```
public class WorkOrderHandler {



  public static void assignTechnicians(List<Work_Order__c> newWorkOrders) {

    // Map to store ServiceType + Location as key to Work Orders

    Map<String, List<Work_Order__c>> criteriaToWorkOrders = new Map<String,
List<Work_Order__c>>();

    List<Work_Order__c> validWorkOrders = new List<Work_Order__c>();



    // Build criteria map from work orders

    for (Work_Order__c wo : newWorkOrders) {

      if (wo.Service_Type__c != null && wo.Location__c != null) {

        String key = wo.Service_Type__c + '::' + wo.Location__c;

        if (!criteriaToWorkOrders.containsKey(key)) {
          criteriaToWorkOrders.put(key, new List<Work_Order__c>());
        }
        criteriaToWorkOrders.get(key).add(wo);
        validWorkOrders.add(wo);
      }
    }

    // Query all available technicians
    List<Technician__c> techs = [SELECT Id, Name, Phone__c, Location__c, Skills__c,
Availability__c, Email__c
                                   FROM Technician__c
                                   WHERE Availability__c = 'Available'];

    List<Assignment__c> assignmentsToInsert = new List<Assignment__c>();

    for (Technician__c tech : techs) {
      String key = tech.Skills__c + '::' + tech.Location__c;
      if (criteriaToWorkOrders.containsKey(key)) {
        List<Work_Order__c> matchedWOs = criteriaToWorkOrders.get(key);
        for (Work_Order__c wo : matchedWOs) {
          Assignment__c assign = new Assignment__c();
          assign.Work_Order_ID__c = wo.Id;
```

```
                assign.Technician_ID__c = tech.Id;

                assignmentsToInsert.add(assign);

            }

            criteriaToWorkOrders.remove(key); // Prevent assigning same WO to multiple
techs

        }

    }



    if (!assignmentsToInsert.isEmpty()) {

        insert assignmentsToInsert;

    }

  }

}
```

2.WorkOrderTrigger(apex)

```
trigger WorkOrderTrigger on WorkOrder__c (after insert) {
   if(trigger.isafter && trigger.isinsert){
      WorkOrderClass.workOrder(trigger.new);
   }
}
```

3. AssigningEmail (apex)

```
public class AssigningEmail {



  public static void sendEmailMsg(List<Assignment__c> assRec) {
    List<Messaging.SingleEmailMessage> emailMessages = new
List<Messaging.SingleEmailMessage>();


    // Query all technicians into a map for easy lookup

    Map<Id, Technician__c> technicians = new Map<Id, Technician__c>(

      [SELECT Id, Phone__c, Location__c, Skills__c, Name__c, Email__c, Availibility__c,
Name

       FROM Technician__c]

    );


    try {

      for (Assignment__c assignment : assRec) {

        if (assignment.Technician_ID__c != null &&
technicians.containsKey(assignment.Technician_ID__c)) {


          Technician__c tech = technicians.get(assignment.Technician_ID__c);


          // Prepare the email

          Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();

          List<String> toAddresses = new List<String>();

          toAddresses.add(tech.Email__c);


          email.setToAddresses(toAddresses);
```

```
                email.setSubject('Work Order Assignment');

                email.setHtmlBody('The following Work Order has been assigned to you.');



                emailMessages.add(email);

              }

            }



            // Send all emails

            if (!emailMessages.isEmpty()) {

                Messaging.sendEmail(emailMessages);

            }

        } catch (Exception e) {

            System.debug('Error -----> ' + e.getMessage());

        }

    }

}
```

4. AssignmentTrigger(apex)

```
trigger AssignmentTrigger on Assignment__c (after insert) {
    if(Trigger.IsAfter && Trigger.IsInsert){
        AssigningEmail.sendEmailmsg(Trigger.New);
    }
}
```

5. CompletionMail(apex)

```
public class CompletionMail {



    public static void sendEmailMsg(List<WorkOrder__c> workOrderList) {

        List<Messaging.SingleEmailMessage> emailMessages = new
List<Messaging.SingleEmailMessage>();



        for (WorkOrder__c wo : workOrderList) {

            if (wo.Status__c == 'Resolved' && String.isNotBlank(wo.Email__c)) {

                Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();



                List<String> toAddresses = new List<String>();

                toAddresses.add(wo.Email__c);



                email.setToAddresses(toAddresses);

                email.setSubject('Status Updated');

                email.setHtmlBody('Your work order has been resolved. Thank you for your
patience.');



                emailMessages.add(email);

            }

        }
        if (!emailMessages.isEmpty()) {

            Messaging.sendEmail(emailMessages);

        }

    }

}
```

6. **WorkOrderTrigger(apex)**

```
trigger WorkOrderTrigger on WorkOrder (before insert, after update) {



    if (Trigger.isBefore && Trigger.isInsert) {

        WorkOrderClass.workOrder(Trigger.new);

    }



    if (Trigger.isAfter && Trigger.isUpdate) {

        CompletionMail.sendEmailMsg(Trigger.new);

    }



}
```

7. **RecordDeletions(apex)**

```
public class RecordDeletions implements Database.Batchable<SObject> {



    public Database.QueryLocator start(Database.BatchableContext bc) {
        String query = 'SELECT Id, Name, WorkOrder_ID__c, Technician_ID__c,
Assignment_Date__c, Completion_Date__c ' +
                       'FROM Assignment__c WHERE Completion_Date__c = LAST_N_DAYS:30';
        return Database.getQueryLocator(query);
    }

    public void execute(Database.BatchableContext bc, List<SObject> scope) {
        List<Assignment__c> assignmentsToDelete = (List<Assignment__c>) scope;

        if (!assignmentsToDelete.isEmpty()) {
            delete assignmentsToDelete;
        }
    }

    public void finish(Database.BatchableContext bc) {
        // Optional: Add notification or logging here
    }
}
```