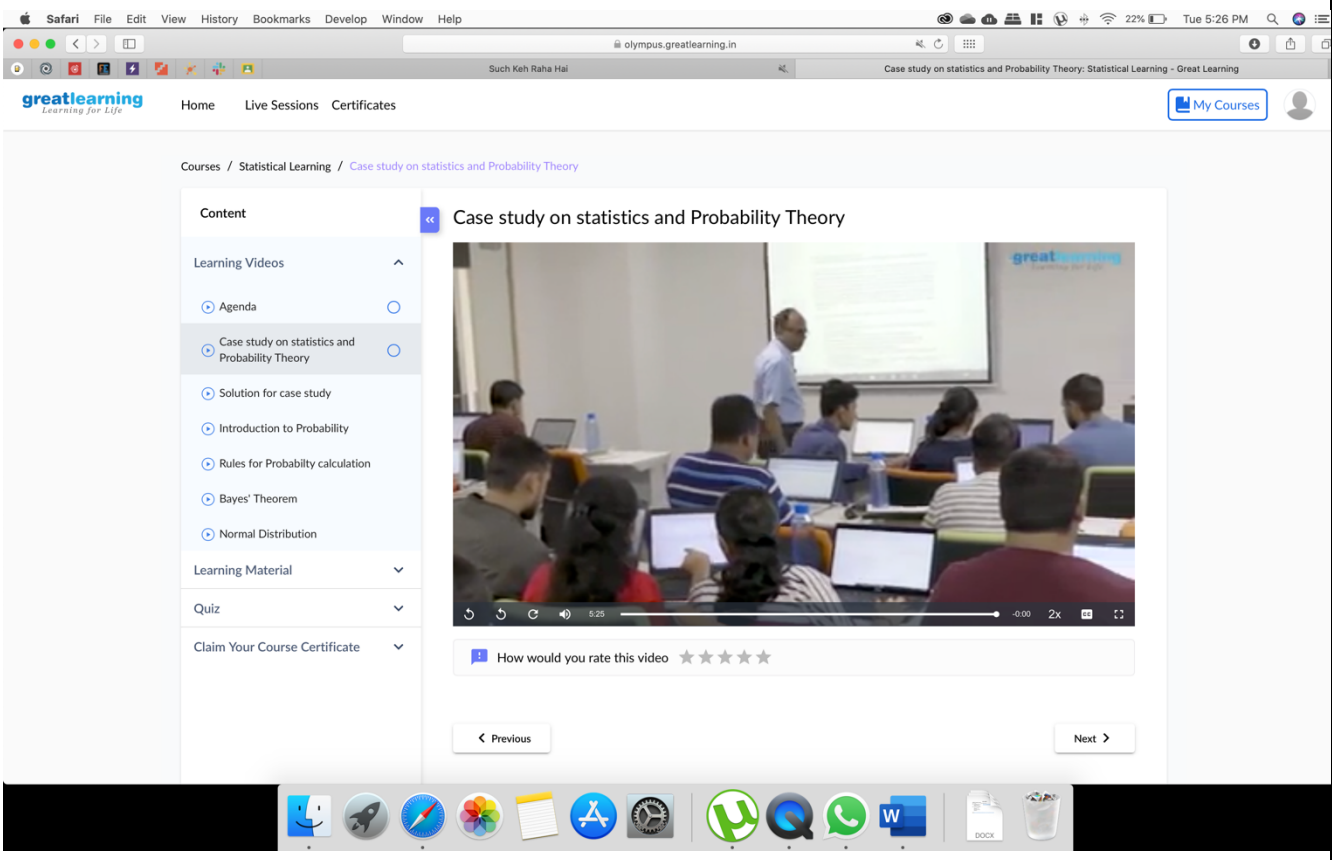
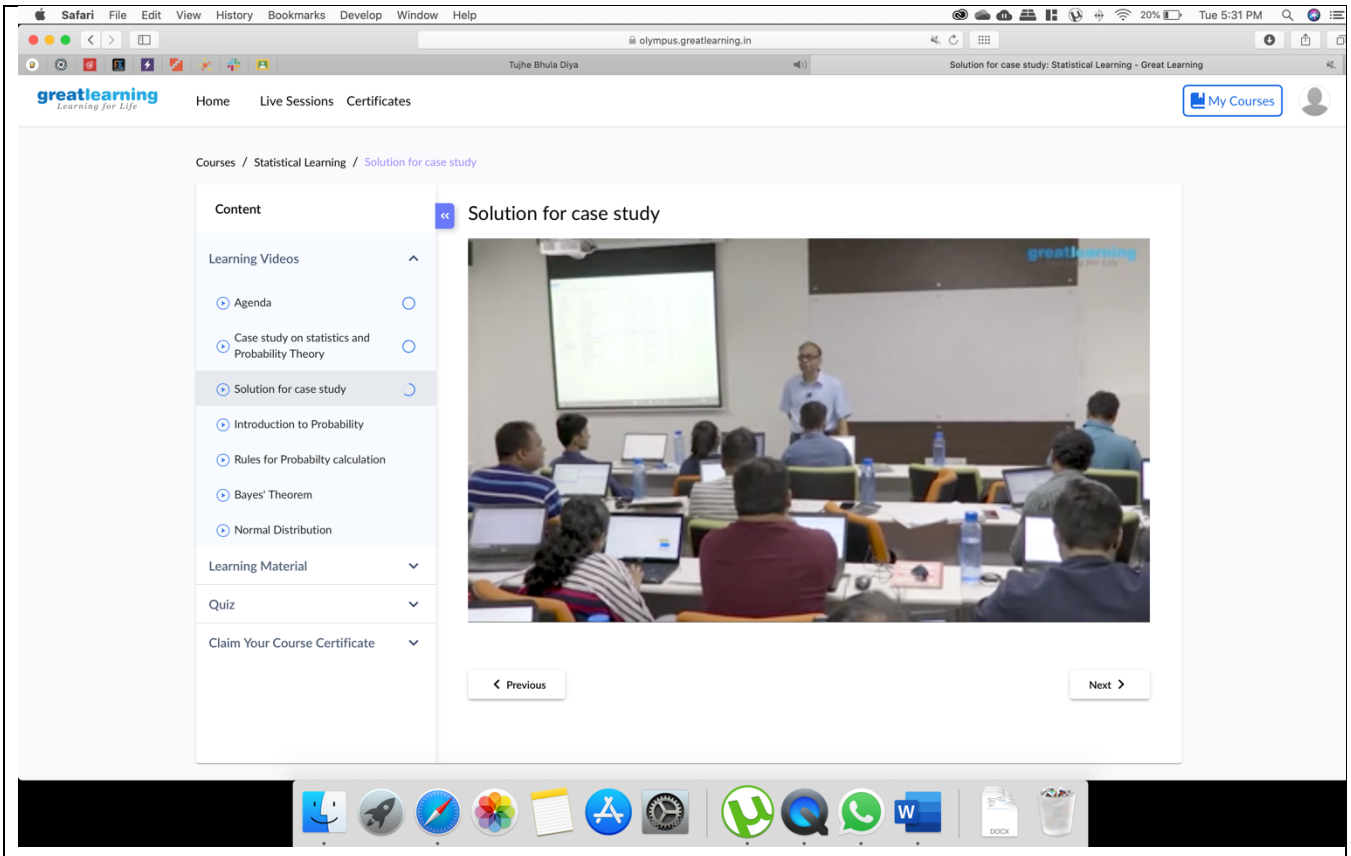
DAILY ASSESSMENT

Date:	**16-06-2020**	**Name:**	**Prajwal Kamagethi Chakravarti P L**
Course:	**Introduction to Digital Marketing**	**USN:**	**4AL17EC073**
Topic:	**Introduction to Digital Marketing**	**Semester & Section:**	**6th & B**
GitHub Repository:	**https://www.github.com/alvas-education-foundation/Prajwal-Kamagethi.git**		

FORENOON SESSION DETAILS

Image of session





Report –

- **The pedagogical advantages of teaching statistics not as a stand-alone subject in itself, but rather as a topic integrated into teaching hands-on, problem-based computer-assisted data analysis.**
- **For over 10 years, such a two-term course has been taught at Drexel University in lieu of the usual statistics courses formerly taken by undergraduate majors in psychology and sociology.**
- **One virtue of the courses as currently implemented is that students seem to learn not just how to perform statistical procedures but how to apply them on their own.**
- **The two-term computer assisted data analysis sequence is also tightly integrated into the department curriculum.**
- **It supports, parallels, and is reinforced by a two-term sequence in research methods, which is often taught by the same instructors. In addition, psychology majors are required to complete a course in experimental psychology.**
- **This integration and coupling of courses afford students who follow the recommended sequence in their sophomore year the marketable skills required for the more desirable cooperative education placements.**

- **It also enables students to engage in reasonable undergraduate research projects later in their education. Some students also elect to go on and do some form of quantitative analysis for their senior thesis, often as part of a faculty member's research.**

Solution for case study:

1. **Read and Examine the Case Thoroughly**
    - **Take notes, highlight relevant facts, underline key problems.**
2. **Focus Your Analysis**
    - **Identify two to five key problems.**
    - **Why do they exist?**
    - **How do they impact the organization?**
    - **Who is responsible for them?**
3. **Uncover Possible Solutions/Changes Needed**

    Review course readings, discussions, outside research, your experience.
4. **Select the Best Solution**

    Consider strong supporting evidence, pros, and cons. Is this solution realistic?

Drafting the Case

Once you have gathered the necessary information, a draft of your analysis should include these general sections, but these may differ depending on your assignment directions or your specific case study:

1. **Introduction**
    - **Identify the key problems and issues in the case study.**
    - **Formulate and include a thesis statement, summarizing the outcome of your analysis in 1–2 sentences.**
2. **Background**
    - **Set the scene: background information, relevant facts, and the most important issues.**

- **Demonstrate that you have researched the problems in this case study.**

3. **Evaluation of the Case**

    - **Outline the various pieces of the case study that you are focusing on.**
    - **Evaluate these pieces by discussing what is working and what is not working.**
    - **State why these parts of the case study are or are not working well.**

4. **Proposed Solution/Changes**

    - **Provide specific and realistic solution(s) or changes needed.**
    - **Explain why this solution was chosen.**
    - **Support this solution with solid evidence, such as:**
        - **Concepts from class (text readings, discussions, lectures)**
        - **Outside research**
        - **Personal experience (anecdotes)**

5. **Recommendations**

    - **Determine and discuss specific strategies for accomplishing the proposed solution.**
    - **If applicable, recommend further action to resolve some of the issues.**
    - **What should be done and who should do it?**

Finalizing the Case

After you have composed the first draft of your case study analysis, read through it to check for any gaps or inconsistencies in content or structure:

- **Is your thesis statement clear and direct?**
- **Have you provided solid evidence?**
- **Is any component from the analysis missing?**

When you make the necessary revisions, <u>proofread and edit</u> your analysis before submitting the final draft.

-

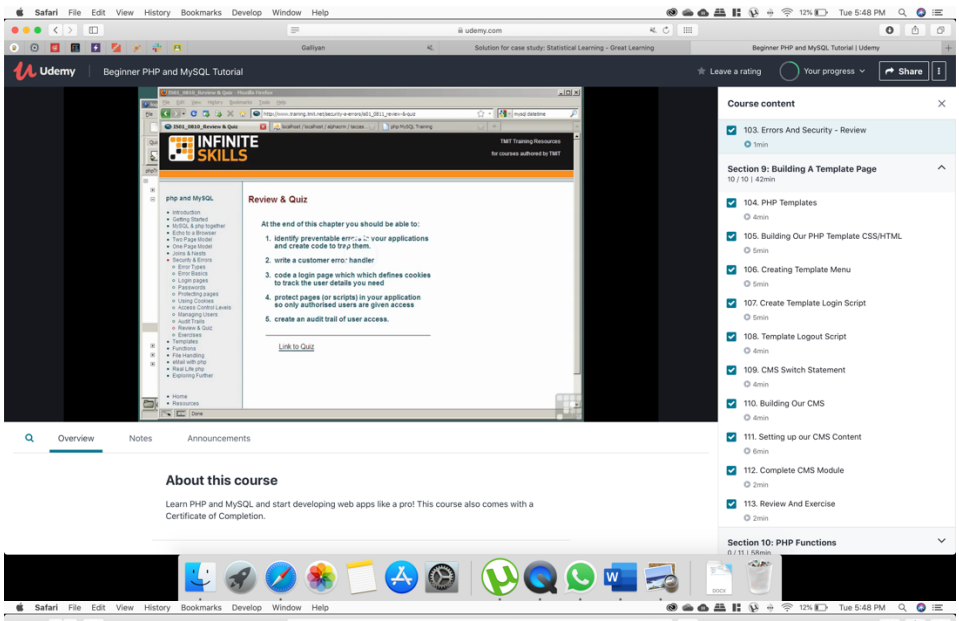
DAILY ASSESSMENT

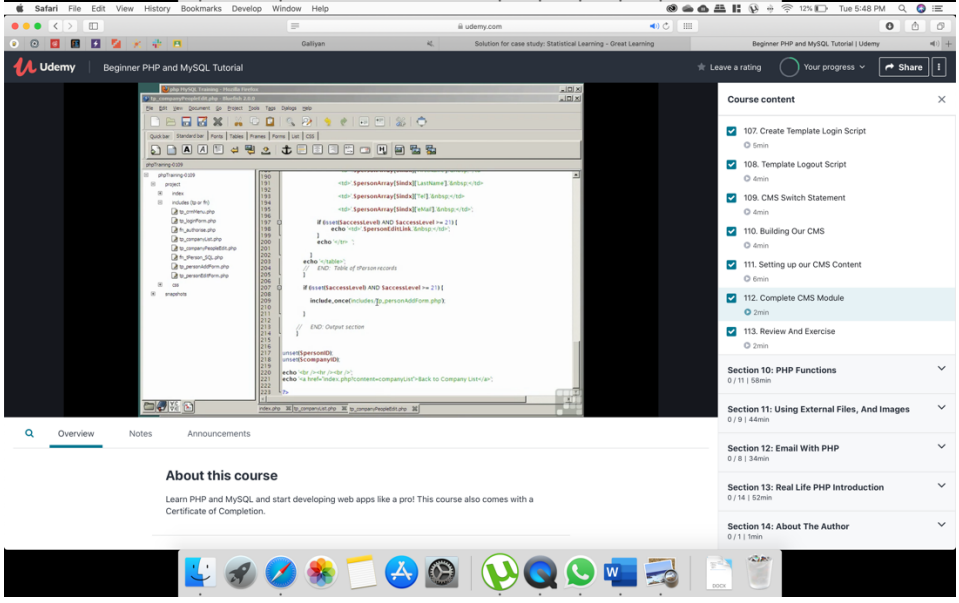
Date:	**16-06-2020**	**Name:**	**Prajwal Kamagethi Chakravarti P L**
Course:	**MySQL**	**USN:**	**4AL17EC073**
Topic:	**1. MySQL Joins** **a. MySQL Join Types - Examples** **b. MySQL Join Diagrams** **c. Creating Nested Lists** **d. MySQL Nested Processing** **e. Styling Our Lists** **f. Intro To Functions** **g. Included Files** **2. PHP Errors and security** **Introduction To Security And Errors** **a. Error Types - Databases** **b. PHP Error Types** **c. Custom Error Testing** **d. Create A Login Page** **e. Passwords For Login** **f. Protecting Pages** **g. Using Cookies - Theory** **h. Using Cookies - Practice** **i. Access Levels - Basic Restriction** **j. Access Levels - Restrict Fields** **k. Managing Users - Structure** **l. Managing Users - Create New** **m. Audit Trails - Access Function** **n. Audit Trails - Logging Logins** **o. Errors And Security – Review** **3. Building a template page** **a. PHP Templates** **b. Building Our PHP Template CSS/HTML** **c. Creating Template Menu** **d. Create Template Login Script**	**Semester &Section:**	**6th & B**

	e. Template Logout Script f. CMS Switch Statement g. Building Our CMS h. Setting up our CMS Content i. Complete CMS Module • Review And Exercise		
GitHub repository	**https://www.github.com/alvas-education-foundation/Prajwal-Kamagethi.git**		

AFTERNOON SESSION DETAILS

Image of session





Report:

With PHP security, there are two sides to error reporting. One is beneficial to increasing security, the other is detrimental.

A standard attack tactic involves profiling a system by feeding it improper data, and checking for the kinds, and contexts, of the errors which are returned. This allows the system cracker to probe for information about the server, to determine possible weaknesses. For example, if an attacker had gleaned information about a page based on a prior form submission, they may attempt to override variables, or modify them:

Example #1 Attacking Variables with a custom HTML page

```
<form method="post" action="attacktarget?username=badfoo&amp;password=badfoo">
<input type="hidden" name="username" value="badfoo" />
<input type="hidden" name="password" value="badfoo" />
</form>
```

The PHP errors which are normally returned can be quite helpful to a developer who is trying to debug a script, indicating such things as the function or file that failed, the PHP file it failed in, and the line number which the failure occurred in. This is all information that can be exploited. It is not uncommon for a php developer to use show_source(), highlight_string(), or highlight_file() as a debugging measure, but in a live site, this can expose hidden variables, unchecked syntax, and other dangerous information. Especially dangerous is running code from known sources with built-in debugging handlers, or using common debugging techniques. If the attacker can determine what general technique you are using, they may try to brute-force a page, by sending various common debugging strings:

Example #2 Exploiting common debugging variables

```
<form method="post" action="attacktarget?errors=Y&amp;showerrors=1&amp;debug=1">
<input type="hidden" name="errors" value="Y" />
<input type="hidden" name="showerrors" value="1" />
<input type="hidden" name="debug" value="1" />
</form>
```

Regardless of the method of error handling, the ability to probe a system for errors leads to providing an attacker with more information.

For example, the very style of a generic PHP error indicates a system is running PHP. If the attacker was looking at an .html page, and wanted to probe for the back-end (to look for known weaknesses in the system), by feeding it the wrong data they may be able to determine that a system was built with PHP.

A function error can indicate whether a system may be running a specific database engine, or give clues as to how a web page or programmed or designed. This allows for deeper investigation into

open database ports, or to look for specific bugs or weaknesses in a web page. By feeding different pieces of bad data, for example, an attacker can determine the order of authentication in a script, (from the line number errors) as well as probe for exploits that may be exploited in different locations in the script.

A filesystem or general PHP error can indicate what permissions the web server has, as well as the structure and organization of files on the web server. Developer written error code can aggravate this problem, leading to easy exploitation of formerly "hidden" information.

There are three major solutions to this issue. The first is to scrutinize all functions, and attempt to compensate for the bulk of the errors. The second is to disable error reporting entirely on the running code. The third is to use PHP's custom error handling functions to create your own error handler. Depending on your security policy, you may find all three to be applicable to your situation.

One way of catching this issue ahead of time is to make use of PHP's own error_reporting(), to help you secure your code and find variable usage that may be dangerous. By testing your code, prior to deployment, with E_ALL, you can quickly find areas where your variables may be open to poisoning or modification in other ways. Once you are ready for deployment, you should either disable error reporting completely by setting error_reporting() to 0, or turn off the error display using the php.ini option *display_errors*, to insulate your code from probing. If you choose to do the latter, you should also define the path to your log file using the *error_log* ini directive, and turn *log_errors* on.

Example #3 Finding dangerous variables with E_ALL

```
<?php
if ($username) {  // Not initialized or checked before usage
   $good_login = 1;
}
if ($good_login == 1) { // If above test fails, not initialized or checked before usage
   readfile ("/highly/sensitive/data/index.html");
}
?>
```