CHAPTER 3

TASKS PERFORMED

3.1 Introduction

The internship was divided into 3 phases one was delta phase which covered the pre-requisites and basics to be learnt, the second phase was interim phase in which me majorly leant about Oracle cloud and its services, the third phase is the final phase which is comprised of learning about technologies and the project

3.2 Delta Phase

In this phase, I was engaged in comprehensive training across a diverse set of technologies integral to modern software development and system integration. This immersive experience encompassed both front-end and back-end development, as well as foundational knowledge in operating systems and data exchange formats.

3.2.1 PL/SQL (Procedural Language/Structured Query Language)

PL/SQL is Oracle's procedural extension of SQL, widely used to create powerful and efficient programs inside the Oracle Database.

Block Components:

Wrote anonymous and named PL/SQL blocks, encapsulating logic to manage transactions. Practiced exception handling for run-time error detection and correction. It comprises of 3 main sections, mainly Declaration, Execution & Exception section as shown in the figure 3.1.

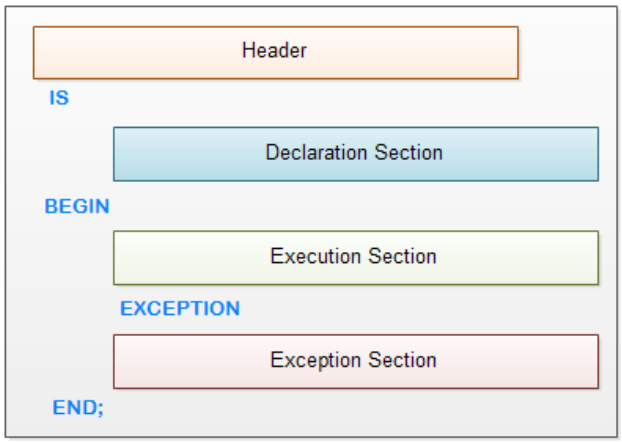

Figure 3.1: Block Structure of PL/SQL

PL/SQL variables naming rules:

Like other programming languages, a variable in PL/SQL must follow the naming rules as follows:

- The variable name must be less than 31 characters. Try to make it as meaningful as possible within 31 characters.
- The variable name must begin with an ASCII letter. It can be either lowercase or uppercase. Notice that PL/SQL is case-insensitive, which means v_data and V_DATA refer to the same variable.
- Followed by the first character are any number, underscore (_), and dollar sign ($) characters. Once again, do not make your variables hard to read and difficult to understand.

PL/SQL Functions:

PL/SQL function is a named block that returns a value. A PL/SQL function is also known as a subroutine or a subprogram. Function may have zero or more than one parameter. You specify the parameter names in the parameter_1, parameter_2, etc. You must specify the data type of each parameter explicitly in the datatype. Each parameter has one of three modes: IN, OUT and IN OUT.

Example:

```
CREATE OR REPLACE FUNCTION try_parse(
iv_number IN VARCHAR2)
RETURN NUMBER IS
BEGIN
RETURN to_number(iv_number);
EXCEPTION
WHEN others THEN
RETURN NULL;
END;
```

PL/SQL Procedures:

Like a PL/SQL function, a PL/SQL procedure is a named block that does a specific task. PL/SQL procedure allows you to encapsulate complex business logic and reuse it in both database layer and application layer. The section before IS keyword is called procedure header or procedure signature.

The elements in the procedure's header are described as follows:

- schema
- name
- parameters
- AUTHID

Example:

```
CREATE OR REPLACE PROCEDURE adjust_salary(
in_employee_id IN EMPLOYEES.EMPLOYEE_ID%TYPE,
in_percent IN NUMBER
) IS
BEGIN
UPDATE employees
SET salary = salary + salary * in_percent / 100
WHERE employee_id = in_employee_id;
END;
```

PL/SQL Exception

In PL/SQL, any kind of errors is treated as exceptions. An exception is defined as a special condition that changes the program execution flow. The PL/SQL provides you with a flexible and powerful way to handle such exceptions.PL/SQL catches and handles exceptions by using exception handler architecture. Whenever an exception occurs, it is raised. The current PL/SQL block execution halts, control is passed to a separate section called exception section.

In the exception section, you can check what kind of exception has been occurred and handle it appropriately. This exception handler architecture enables separating the business logic and exception handling code hence make the program easier to read and maintain.

There are two types of exceptions:

- **System exception**: the system exception is raised by PL/SQL run-time when it detects an error. For example, NO_DATA_FOUND exception is raised if you select a non-existing record from the database.
- **Programmer-defined exception**: the programmer-defined exception is defined by you in a specific application. You can map exception names with specific Oracle errors using the EXCEPTION_INIT pragma. You can also assign a number and description to the exception using RAISE_APPLICATION_ERROR.

Example:

```
EXCEPTION
WHEN BELOW_SALARY_RANGE THEN
dbms_output.put_line('Employee ' || n_emp_id ||' has salary below the salary range');
WHEN ABOVE_SALARY_RANGE THEN
dbms_output.put_line('Employee ' || n_emp_id ||' has salary above the salary range');
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('Employee ' || n_emp_id || ' not found');
END;
/
```

PL/SQL Cursor

A PL/SQL cursor is a pointer that points to the result set of an SQL query against database tables. To use PL/SQL cursor, first you must declare it in the declaration section of PL/SQL block or in a package as follows:

```
CURSOR cursor_name [ ( [ parameter_1 [, parameter_2 ...] ) ]
[ RETURN return_specification ]
IS sql_select_statements
[FOR UPDATE [OF [column_list]];
```

First, you declare the name of the cursor cursor_nameafter the CURSOR keyword. The name of the cursor can have up to 30 characters in length and follows the naming rules of identifiers in PL/SQL. It is important to note that cursor's name is not a variable so you cannot use it as a variable such as assigning it to other cursor or using it in an expression. The parameter1, parameter2… are optional elements in the cursor declaration. These parameters allow you to pass arguments into the cursor. The RETURN return_specification is also an optional part.

Second, you specify a valid SQL statement that returns a result set where the cursor points to.Third, you can indicate a list of columns that you want to update after the FOR UPDATE OF. This part is optional so you can omit it in the CURSOR declaration.

PL/SQL Triggers

Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events –

- Database manipulation (DML) statement (DELETE, INSERT, or UPDATE)
- Database definition (DDL) statement (CREATE, ALTER, or DROP).
- Database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

Example:

CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF}
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name] ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE // Declaration-statements.
BEGIN // Executable-statements.
EXCEPTION // Exception-handling-statements
END.

3.2.2 HTML5, CSS3, JS

This module emphasized both design and interactivity of web pages, laying the foundation for responsive, user-friendly interfaces. Integrated media using <audio>, <video>, and interactive drawing tools with <canvas>, including custom charts and shapes using JavaScript.

HTML5

HTML is a standard markup language, which stands for Hyper Text Markup Language. It is widely used language to create webpages. HTML invented by Tim Berners-Lee in late 1991, but it's first version "HTML 1.0" was released in 1993, and "HTML 2.0" was the first standard HTML specification, which was published in 1995.

HTML Media

The <video> element is used to enable video playback support within a web page. It works very similarly to the <img> element, as it also requires adding the path or URL of the

video within the src attribute. The HTML supports only MP4, WebM, and Ogg video formats. The <video> element also supports audio; however, the <audio> element is more suitable for that purpose.

Example:

```
<video width="640" height="360" controls>
  <source src="video-file.mp4" type="video/mp4">
  <source src="video-file.ogg" type="video/ogg">
  Your browser does not support the video tag.
</video>
```

The <audio> element is used to enable the support of audio files within a web page. We can include multiple sources of audio; however, the browser will choose the most appropriate file automatically. Most of the attributes of <video> element is also compatible with the <audio> element. The most frequently used attributes of the HTML audio element are controls, autoplay, loop, muted, and src.

Example:

```
<audio controls>
  <source src="file_path" type="audio/mpeg">
  Your browser does not support the audio element.
</audio>
```

HTML Canvas

HTML element <canvas> gives you an easy and powerful way to draw graphics using JavaScript. It can be used to draw graphs, make photo compositions or do simple (and not so simple) animations.

Example:

```
<!DOCTYPE html>
<html><head>
 <style>
     #mycanvas{border:1px solid red;}
   </style>
</head>
<body>
<canvas id="mycanvas" width="100" height="100"></canvas>
script type="text/javascript">
```

```
    function drawShape(){
        var canvas = document.getElementById('mycanvas');
        if (canvas.getContext){
        var ctx = canvas.getContext('2d');
        ctx.fillRect(25,25,100,100);
        ctx.clearRect(45,45,60,60);
        ctx.strokeRect(50,50,50,50);
      } else
        alert('You need Safari or Firefox 1.5+ to see this demo.');
    }
  </script>
</body></html>
```

CSS3

CSS is the acronym for "Cascading Style Sheet". It's a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS helps the web developers to control the layout and other visual aspects of the web pages. CSS plays a crucial role in modern web development by providing the tools necessary to create visually appealing, accessible, and responsive websites.

CSS3 (Cascading Style Sheets Level 3)

CSS3 is a collection of modules that extend the capabilities of CSS. It introduces numerous new features and enhancements, including advanced selectors, multiple column layouts, animations, transformations, gradients, shadows, and more.

Types of CSS

CSS stands for "Cascading Style Sheet". It is a style sheet language used to control the layout and other visual aspects of the web pages. There are three types of CSS which are mentioned below:

- **Inline CSS:** Inline CSS is applied directly to an HTML element using the style attribute.
- **Internal CSS:** Internal CSS is defined within the <style> tag inside the <head> section of an HTML document.
- **External CSS:** External CSS is written in a separate .css file and linked to the HTML document using the <link> tag.

CSS Fonts

In CSS, the font property is used to style and adjust type of text used in webpage. You can define fonts and customize their appearance by setting properties like font-family, font-size, font-weight and font-style. You can also use the shorthand property font to manipulate all the font style**.**

Types of CSS Fonts

- **Monospace Fonts**: The font in which every letter have equal width.
- **Serif Fonts**: Have small stroke at the edge of each letter.
- **San-Serif Fonts**: Clean fonts without any strokes.
- **Fantasy Fonts**: Decorative fancy fonts.
- **Cursive Fonts**: The font that resembles human handwriting.

CSS Tables

Styling tables in a webpage involves using CSS properties to customize the appearance of tables. CSS properties such as border-collapse, border-spacing, and caption-side can be applied to tables to control the borders, spacing, and alignment of the table and its cells.

To style table borders, we use CSS properties like border and border-radius. You can set the border's width, color, and style with border property on the table, rows, or individual cells.

- **border:** CSS border property sets the width, style, and color of all four sides of the table border (e.g., border: 1px solid black;).
- **border-radius:** CSS border-radius property rounds the corners of the table border (e.g., border-radius: 5px|50%).

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    table {
      border-radius: 10px;
      border: 2px solid #031926;
      width: 100%;
    }
    td {
```

```
        border: 1px solid black;
    }
  </style>
</head>
<body>
  <table>
    <tr>
      <th>Header 1</th>
    </tr>
    <tr>
      <td> Data 1</td>
    </tr>
  </table>
</body>
</html>
```

CSS Hover

CSS hover effects are used to make interactive elements such as buttons and links more interactive. The :hover pseudo-class in CSS is used to target an element when the user hovers over it with the mouse cursor. Its purpose is to apply styles to improve the user experience. The :hover property can be used in various scenarios such as to change the button color when we hover it, adjust the size of div boxes, or show the hidden content.

In CSS, the pseudo-class :hover is a type of selector used to target and style an element when a user moves the mouse pointer over the element.Hover effects are mostly used with interactive elements like buttons, links, etc to provide the user with a dynamic experience. Hover effects are useful to add a dynamic and engaging look to a website.

Example:

```
.container{
        width: 40%;
        height: 100px;
        background-color: #D5E8D4;
    }
.container: hover{
```

```
    background-color: #33FF33;
}
```

CSS Transforms

CSS transforms are used to modify the element's shape and sizes and are responsible for movements of elements in two-dimensional space using functions like translate (), scale(), rotate(), and skew(). These functions allow you to move, scale, rotate, and skew elements along the X and Y axes, creating various visual effects and manipulations. Types of transformations are:

- CSS Translate
- CSS Rotate
- CSS Scale
- CSS Skew

CSS Transitions

CSS transition property allows you to animate changes in an element's style properties over a specified duration. They provide a simple and efficient way to add animations to web elements without the need for complex JavaScript code or external libraries.CSS transition property is a shorthand property for:

- transition-property
- transition-duration
- transition-timing-function
- transition-delay
- transition-behavior

CSS Animations

In CSS we can dynamically change styles of elements based on time duration, user interaction or state changes called CSS animations. It is implemented using the `@keyframes` rule to create the animation and the animation property to apply it to an element.

Example:

```
.ball {
    border-radius: 50%;
    animation-name: moveRight ;
    animation-duration: 2s;
}
```

```
@keyframes moveRight {
        to {
            left: calc(100% - 50px);
             }
 }
```

JavaScript

JavaScript is a lightweight, interpreted programming language. It is commonly used to create dynamic and interactive elements in web applications. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.

Features

- JavaScript is the most popular programming language in the world, making it a programmer's great choice. Once you learn JavaScript, it helps you develop great front-end and back-end software using different JavaScript based frameworks like jQuery, Node.JS, etc.
- JavaScript is everywhere, it comes installed on every modern web browser and so to learn JavaScript, you really do not need any special environment setup. For example, Chrome, Mozilla Firefox, Safari, and every browser you know as of today, supports JavaScript.
- JavaScript helps you create really beautiful and crazy fast websites. You can develop your website with a console like look and feel and give your users the best Graphical User Experience.
- JavaScript usage has now extended to mobile app development, desktop app development, and game development. This opens many opportunities for you as JavaScript Programmer.
- Due to high demand, there is tons of job growth and high pay for those who know JavaScript. You can navigate over to different job sites to see what having JavaScript skills looks like in the job market.
- Furthermore, JavaScript has more than 1.5 lakh libraries. It is also growing.
- A huge community of JavaScript is available on the internet with students, developers, and mentors. So, anyone can easily get support.

3.2.3 RESTful Webservices

RESTful Web Services are basically REST Architecture based Web Services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications. This tutorial will teach you the basics of RESTful Web Services and contains chapters discussing all the basic components of RESTful Web Services with suitable examples.

Resources

REST architecture treats every content as a resource. These resources can be Text Files, Html Pages, Images, Videos or Dynamic Business Data. REST Server simply provides access to resources and REST client accesses and modifies the resources. Here each resource is identified by URIs/ Global IDs. REST uses various representations to represent a resource where Text, JSON, XML. The most popular representations of resources are XML and JSON.

Example:

<user>
<id>1</id>
<name>Mahesh</name>
<profession>Teacher</profession>
</user>

HTTP Request

RESTful Web Services make use of HTTP protocols as a medium of communication between client and server.

An HTTP Request has five major parts as shown in figure 3.2 −

- **Verb**: Indicates the HTTP methods
- **URI:** identify the resource on the server.
- **HTTP Version:** Indicates the HTTP version. For example, HTTP v1.1.
- **Request Header:** Contains metadata for the HTTP Request message as key-value pairs.
- **Request Body:** Message content or Resource representation.

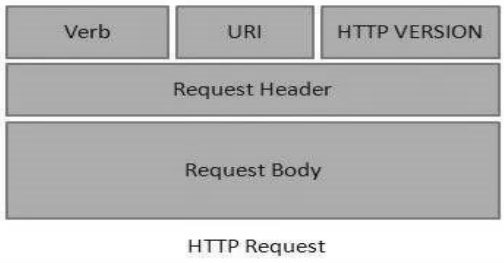

Figure 3.2: HTTP Request

HTTP Response

An HTTP Response has four major parts as shown in figure 3.3 −

- **Status/Response Code:** Indicates the Server status for the requested resource.
- **HTTP Version**: Indicates the HTTP version.
- **Response Header**: Contains metadata for the HTTP Response message as key value pairs.
- **Response Body**: Response message content or Resource representation.

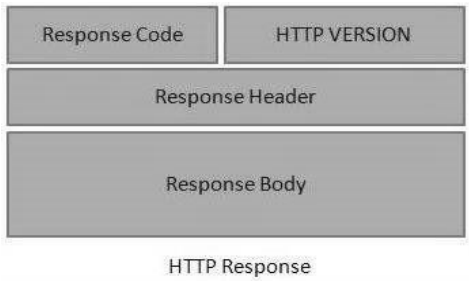

Figure 3.3: HTTP Response

Statelessness

As per the REST architecture, a RESTful Web Service should not keep a client state on the server. This restriction is called Statelessness. It is the responsibility of the client to pass its context to the server and then the server can store this context to process the client's further request. For example, session maintained by server is identified by session identifier passed by the client.

Caching

Caching refers to storing the server response in the client itself, so that a client need not make a server request for the same resource again and again. A server response should have information about how caching is to be done, so that a client caches the response for a time-period or never caches the server response.

HTTP Code

As RESTful Web Services work with HTTP URL Paths, it is very important to safeguard a RESTful Web Service in the same manner as a website is secured.

200: OK − shows success.

201: CREATED − when a resource is successfully created using POST or PUT request.

401: UNAUTHORIZED − states that user is using invalid or wrong authentication token.

404: NOT FOUND − states that the method is not available.

500: INTERNAL SERVER ERROR − states that the server has thrown some exception while executing the method.

3.2.4 XML (eXtensible Markup Language)

XML is widely used for data exchange in web services and configurations. This module focused on understanding XML structure, validation, and parsing techniques. XML stands for Extensible Markup Language and is a text-based markup language derived from Standard Generalized Markup Language (SGML). This tutorial will teach you the basics of XML.

There are three important characteristics of XML that make it useful in a variety of systems and solutions −

- **XML is extensible**: XML allows you to create your own self-descriptive tags, or language, that suits your application.
- **XML carries the data, does not present it:** XML allows you to store the data irrespective of how it will be presented.
- **XML is a public standard**: XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

Following syntax shows XML declaration −

```
<?xml
version = "version_number"
encoding = "encoding_declaration"
standalone = "standalone_status"
?>
```

XML Validations

Validation is a process by which an XML document is validated. An XML document is said to be valid if its contents match with the elements, attributes and associated document

type declaration(DTD), and if the document complies with the constraints expressed in it. Validation is dealt in two ways by the XML parser. They are –

- Well-formed XML document
- Valid XML document

XML DTD's

The XML Document Type Declaration, commonly known as DTD, is a way to describe XML language precisely. DTDs check vocabulary and validity of the structure of XML.
An XML DTD can be either specified inside the document, or it can be kept in a separate document and then liked separately.

SYNTAX:

<!DOCTYPE element DTD identifier
[declaration1, declaration2.....]>

XML Schemas

XML Schema is commonly known as XML Schema Definition (XSD). It is used to describe and validate the structure and the content of XML data. XML schema defines the elements, attributes and data types. Schema element supports Namespaces. It is similar to a database schema that describes the data in a database.

Example

<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
<xs:element name = "contact">
<xs:complexType>
<xs:sequence>
<xs:element name = "name" type = "xs:string" />
<xs:element name = "company" type = "xs:string" />
<xs:element name = "phone" type = "xs:int" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

XML Tree Structure

In the figure 3.4, there is a root element named as <company>. Inside that, there is one more element <Employee>. Inside the employee element, there are five branches named

<FirstName>, <LastName>, <ContactNo>, <Email>, and <Address>. Inside the <Address> element, there are three sub-branches, named <City> <State> and <Zip>.

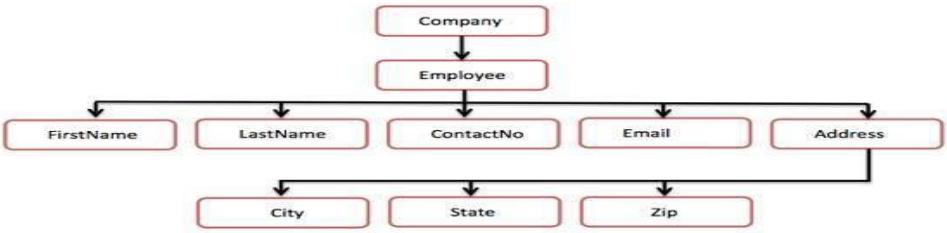

Figure 3.4: Tree Structure

XML DOM

The Document Object Model (DOM) is the foundation of XML. XML documents have a hierarchy of informational units called nodes; DOM is a way of describing those nodes and the relationships between them. A DOM document is a collection of nodes or pieces of information organized in a hierarchy. This hierarchy allows a developer to navigate through the tree looking for specific information. Because it is based on a hierarchy of information, the DOM is said to be tree based. The XML DOM, on the other hand, also provides an API that allows a developer to add, edit, move, or remove nodes in the tree at any point in order to create an application.

XML Parsers

XML parser is a software library or a package that provides interface for client applications to work with XML documents. It checks for proper format of the XML document and may also validate the XML documents as shown in the figure 3.5. Modern day browsers have built-in XML parsers.

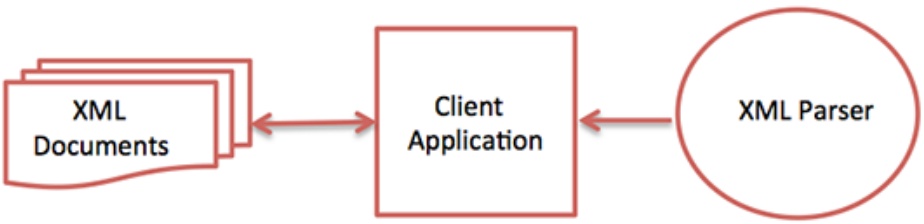

Figure 3.5: Parser Interactions

XML Processors

When a software program reads an XML document and takes actions accordingly, this is called processing the XML.

An XML processor reads the XML file and turns it into in-memory structures that the rest of the program can access. The most fundamental XML processor reads an XML document and converts it into an internal representation for other programs or subroutines to use. This is called a parser, and it is an important component of every XML processing program.

3.2.5 Unix Shell Scripting

The Linux operating system is a set of programs which acts as a link between the computer and the end user. The computer programs that allocate the system resources and coordinate all the details of the computer's internals is called the Operating System or the Kernel. UNIX Shell Scripting as shown in figure 3.6 is a critical skill for automating backend tasks and interacting with server environments.

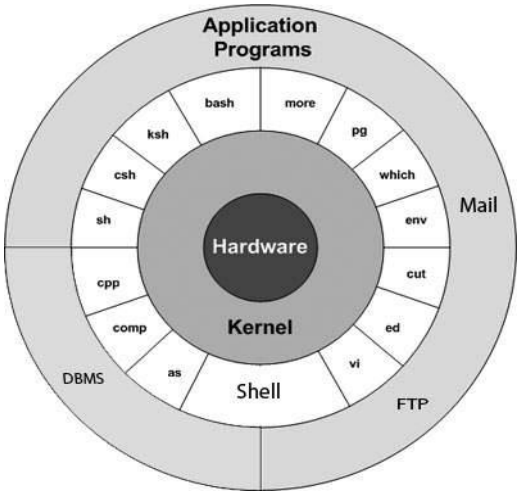

Figure 3.6 Block Diagram of Unix

- **Kernel:** The kernel is the heart of the Linux operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.
- **Shell:** The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are the most famous shells which are available with most of the Unix variants.

- **Commands and Utilities**: There are various commands and utilities which you can make use of in your day-to-day activities. ftp, ssh, cp, mv, cat and grep, etc. are
  There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various options.
- **Files and Directories**: All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the filesystem.

Shell Scripts

The basic concept of a shell script is a list of commands as shown in the table 3.1, which are listed in the order of execution. A good shell script will have comments, preceded by # sign, describing the steps.

There are conditional tests, such as value A is greater than value B, loops allowing us to go through massive amounts of data, files to read and store data, and variables to read and store data, and the script may include functions.

Table 3.1: Basic commands used in Unix shell scripting.

Command	Description
ls	Lists files and directories in the current directory.
ls -l	Lists in long format (with permissions, size, date, etc.).
pwd	Prints the current working directory.
cd [directory]	Changes the current directory to the specified one.
mkdir [directory]	Creates a new directory.
rmdir [directory]	Removes an empty directory.
rm [file]	Deletes a file.
cp [source] [destination]	Copies files or directories.
mv [source] [destination]	Moves or renames files or directories.
touch [file]	Creates a new empty file.
cat [file]	Displays the content of a file.

File Management

All the data in Linux is organized into files and all these files are organized into different directories. These directories are organized into a tree-like structure called the filesystem. As such everything in Linux is a file. So, when you work with Unix, one way or another, you spend most of your time working with files. In Unix, there are three basic types of files −

- **Ordinary Files:** An ordinary file is a file on the system that contains data, text, or program instructions. In this tutorial, you look at working with ordinary files.
- **Directories**: Directories store both special and ordinary files. For users familiar with Windows or Mac OS, Unix directories are equivalent to folders.
- **Special Files**: Some special files provide access to hardware such as hard drives, CD-ROM drives, modems, and Ethernet adapters. Other special files are like aliases or shortcuts and enable you to access a single file using different names.

File Permissions

File ownership is an important component of Unix that provides a secure method for storing files. Every file in Unix has the following attributes −

- **Owner permissions**: The owner's permissions determine what actions the owner of the file can perform on the file.
- **Group permissions**: The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
- **Other (world) permissions**: The permissions for others indicate what action all other users can perform on the file.

File Access Modes

The permissions of a file are the first line of defense in the security of a Unix system. The basic building blocks of Unix permissions are the read, write, and execute permissions, which have been described below −

- **Read:** Grants the capability to read, i.e., view the contents of the file.
- **Write:** Grants the capability to modify or remove the content of the file.
- **Execute:** User with execute permissions can run a file as a program.

Using chmod with symbolic mode

The easiest way for a beginner to modify file or directory permissions is to use the symbolic mode. With symbolic permissions you can add, delete, or specify the permission set you want by using the operators in the following table 3.2.

Table 3.2: Operators of Chmod in symbolic mode

Sl.No.	Chmod operator & Description
1	(+) Adds the designated permission(s) to a file or directory.
2	(-) Removes the designated permission(s) from a file or directory.
3	(=) Sets the designated permission(s).

Using chmod with Absolute Permissions

Table 3.3: Octal Representation in absolute Permissions

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

The second way to modify permissions with the chmod command is to use a number to specify each set of permissions for the file as show in above table 3.3. Each permission is assigned a value, as the following table shows, and the total of each set of permissions provides a number for that set.

3.3 Interim Phase

3.3.1 OCI (Oracle Cloud Infrastructure)

The cloud refers to servers that are accessed over the Internet, and the software and databases that run on those servers. Cloud servers are located in data centers all over the world. Cloud infrastructure refers to the resources needed for hosting and building applications in the cloud. IaaS and PaaS services are often included in an organization's cloud infrastructure, although SaaS can be said to be part of cloud infrastructure as well, and FaaS offers the ability to construct infrastructure as code.

Types of Cloud

The most common cloud deployments are:

- **Private cloud:** A private cloud is a server, data center, or distributed network wholly dedicated to one organization.
- **Public cloud:** A public cloud is a service run by an external vendor that may include servers in one or multiple data centers. Unlike a private cloud, public clouds are shared by multiple organizations. Using virtual machines, individual servers may be shared by different companies, a situation that is called "multitenancy" because multiple tenants are renting server space within the same server.
- **Hybrid cloud:** hybrid cloud deployments combine public and private clouds and may even include on-premises legacy servers. An organization may use their private cloud for some services and their public cloud for others, or they may use the public cloud as backup for their private cloud.
- **Multi-cloud:** multi-cloud is a type of cloud deployment that involves using multiple public clouds. In other words, an organization with a multi-cloud deployment rents virtual servers and services from several external vendors — to continue the analogy used above, this is like leasing several adjacent plots of land from different landlords. Multi-cloud deployments can also be hybrid cloud, and vice versa.

Service Models

The resources available in the cloud are known as "services," since they are actively managed by a cloud provider. Cloud services include infrastructure, applications, development tools, and data storage, among other products.

These services are sorted into several different categories, or service models. Types of service models includes:

- **Software-as-a-Service (SaaS):** Instead of users installing an application on their device, SaaS applications are hosted on cloud servers, and users access them over the Internet. SaaS is like renting a house: the landlord maintains the house, but the tenant mostly gets to use it as if they owned it. Examples of SaaS applications include Salesforce, MailChimp, and Slack.
- **Platform-as-a-Service (PaaS):** In this model, companies don't pay for hosted applications; instead they pay for the things they need to build their own applications. PaaS vendors offer everything necessary for building an application, including development tools, infrastructure, and operating systems, over the Internet. PaaS can be compared to renting all the tools and equipment necessary for building a house, instead of renting the house itself. PaaS examples include Heroku and Microsoft Azure.
- **Infrastructure-as-a-Service (IaaS):** In this model, a company rents the servers and storage they need from a cloud provider. They then use that cloud infrastructure to build their applications. IaaS is like a company leasing a plot of land on which they can build whatever they want — but they need to provide their own building equipment and materials. IaaS providers include DigitalOcean, Google Compute Engine, and OpenStack.

OCI Components

Compute: Compute refers to the virtual machines (VMs), bare metal servers, or container-based environments that run your applications and workloads.

Key Features include:

- **VMs (Virtual Machines):** A virtualized server offering flexible configurations (e.g., CPU, memory, storage) for general compute needs.
- **Bare Metal:** High-performance servers with no virtualization layer, ideal for workloads that require dedicated resources.

- **Oracle Container Engine for Kubernetes (OKE):** Managed Kubernetes service for containerized applications.
- **Functions**: Serverless compute for running functions in response to events.

Storage: Storage refers to the different options for saving data – whether it's files, block storage for VMs, or object storage for backups and unstructured data. OCI offers various storage options to cater to different use cases.

- **Block Volumes**: High-performance, durable storage volumes for use with compute instances (VMs).
- **Object Storage**: A scalable and durable storage service for storing large amounts of unstructured data, including backups and media files.
- **File Storage**: Managed NFS service for applications that require shared file systems.
- **Archive Storage**: Low-cost, long-term storage for rarely accessed data.
- **Local NVMe Storage**: Fast storage for high-performance compute workloads.

Networking: Networking provides the connectivity between your resources in OCI and to the internet or on-premises systems. OCI provides flexible networking solutions for cloud-based architectures:

- **Virtual Cloud Network (VCN):** An isolated network for connecting cloud resources.
- **Subnets**: Logical subdivisions within a VCN to organize resources and control traffic.
- **Load Balancing**: Distributes incoming traffic across multiple instances
- **DNS**: Managed DNS services for domain name resolution.
- **VPN**: Secure connection between on-premises networks and OCI.

Identity and Access Management (IAM): OCI IAM enables secure control over who can access your cloud resources and what actions they can perform. IAM in OCI helps manage users, groups, and policies to control access to resources.

- **Users**: Individuals or systems that interact with OCI resources.
- **Groups**: Collections of users with common access requirements.
- **Policies**: Rules that define what actions groups or users can perform on OCI resources.
- **Federation:** Integrates external identity providers for single sign-on.
- **Dynamic Groups**: Groupings based on resource attributes.

Database: OCI offers a range of databases to suit various workloads — from simple apps to complex, data-heavy enterprise systems**.** OCI provides several types of databases optimized for both transactional (OLTP) and analytical workloads (OLAP):

- **Autonomous Database**: A self-managing, self-patching, and self-tuning database.
- **Oracle Database (VM/Bare Metal):** Traditional database service on OCI, including features such as high availability and disaster recovery.
- **Oracle MySQL**: Managed MySQL database service.
- **NoSQL Database:** A schema-less, horizontally scalable database for large volumes of unstructured data.

Security: Security is a shared responsibility between Oracle and the customer. OCI provides built-in tools and features to secure cloud environments. Security in OCI is built into the core infrastructure and service offerings:

- **Cloud Guard:** Provides security posture management by identifying and mitigating misconfigurations.
- **Vault**: Manages and secures encryption keys and secrets for applications.
- **Web Application Firewall (WAF):** Protects applications from common web threats (e.g., DDoS, SQL injection).
- **Identity & Access Management (IAM)**: Ensures that users and services have the appropriate permissions.

OIC Workflow in OCI

Oracle Integration Cloud (OIC) provides a powerful platform for building, automating, and managing integration workflows.

Pre-built Connectors: For integrating with Oracle and third-party applications (e.g., Oracle ERP, Salesforce, and ServiceNow).

Process Automation: Define workflows for business processes, e.g., order-to-cash, procure-to-pay, etc.

Real-time Monitoring: Built-in monitoring tools to track process execution and errors.

Error Handling: Built-in error handling and retry capabilities in the workflow.

Monitoring and Observability

OCI offers comprehensive monitoring capabilities:

- **OCI Monitoring Service**: Provides metrics collection, alerting, and dashboards.
- **Logs and Events**: Allows you to collect, store, and analyze logs and events from various resources.

- **Application Performance Monitoring (APM):** Monitors the performance of your applications to identify bottlenecks and optimize user experiences.
- **Resource Utilization Metrics:** Provides data on CPU, memory, and storage usage for resources, enabling cost management.

3.3.2 Oracle Integration Cloud (OIC)

In Oracle Integration Cloud (OIC), an integration refers to a process that connects two or more applications, systems, or services so they can exchange data and perform coordinated tasks. It acts as a bridge between different endpoints (like ERP, CRM, databases, cloud services, etc.), enabling seamless data flow and business process automation.

OIC provides a unified platform that includes application integration, process automation, visual application building, and real-time analytics. It is built to support both business and IT users by offering low-code tools, prebuilt integrations, and rich connectivity options. OIC is designed to handle hybrid environments, integrating SaaS, PaaS, and on-premises applications with minimal development effort.

Cloud Architecture

OIC's architecture is multi-tenant and cloud-native, ensuring scalability, reliability, and high availability. It comprises several key components:

- **Integration Layer:** For designing and executing integrations.
- **Process Automation Layer**: For business workflows and human task flows.
- **Visual Builder:** For rapid web and mobile application development.
- **Data Layer**: For handling lookups, variables, and connections.
- **Monitoring Layer:** Provides dashboards and analytics for tracking performance and errors.

Integration Scenario

A typical integration scenario using OIC involves**:**

- Connecting an ERP system like Oracle Fusion with third-party systems such as Salesforce.
- Automating order-to-cash process, where an order placed in a CRM triggers a corresponding sales order in ERP.
- Sending real-time updates to downstream systems like inventory or billing modules.

Features & Benefits

- **Low-Code Development**: Visual interfaces and drag-and-drop tools.
- **Prebuilt Integrations**: Hundreds of prebuilt adapters and templates.
- **Scalability**: Handles high data volumes efficiently.
- **Real-Time Monitoring**: Dashboards for proactive management.
- **Hybrid Connectivity**: Bridges cloud and on-premises environments.
- **Security**: Secure transmission using encryption and authentication protocols.

OIC Components

- **Integrations**: Logic and data mapping between systems.
- **Connections**: Authentication and configuration to external systems.
- **Lookups**: Mapping tables for data transformations.
- **Agents**: Allow secure on-premises connectivity.
- **Libraries**: Reusable artifacts across integrations.
- **Packages**: Bundles of integrations, connections, and related artifacts.

OIC Adapters

Oracle Integration Cloud (OIC) offers a wide variety of adapters that simplify connectivity with cloud, on-premises, and third-party applications. These adapters abstract the complexities of different protocols and data formats, enabling seamless integration. Different types of adapters:

Application Adapters: These adapters connect to popular SaaS and enterprise applications. They support business objects, services, and operations specific to each application.

- **Oracle ERP Cloud Adapter**: Integrates with Oracle Fusion Applications for automating finance, procurement, and HCM workflows. Supports business events, FBDI, and SOAP/REST services.
- **Salesforce Adapter**: Enables connectivity to Salesforce objects (like Account, Contact). It supports SOQL queries, CRUD operations, and event-based triggers.
- **ServiceNow Adapter**: Automates workflows in ServiceNow (e.g., ticketing systems) by interacting with incident, change request, and other modules.
- **Oracle HCM Cloud Adapter**: Integrates HR processes by connecting to Oracle HCM, providing access to worker data, job requisitions, payroll, etc.

Technology Adapters: These adapters handle standard technical protocols and services like REST, SOAP, FTP, database operations, and files.

- **REST Adapter**: Allows you to invoke or expose RESTful APIs. You can configure resources, methods (GET, POST, PUT, DELETE), query parameters, headers, and security.
- **SOAP Adapter**: Integrates with SOAP-based web services using WSDL. It supports custom headers, fault handling, and XML payloads.
- **FTP Adapter**: Used for reading/writing files to/from FTP/SFTP servers. Ideal for batch processing scenarios.
- **File Adapter**: Operates on local file systems or mount points. Common in on-premises integrations via Connectivity Agent.
- **Database Adapter (DB Adapter):** Connects to relational databases (Oracle, MySQL, etc.) for performing SQL operations like SELECT, INSERT, UPDATE, DELETE, and invoking stored procedures.

Cloud Storage Adapters: These are used for integrating cloud-based storage services, which are often used in document management and data archival.

- **Oracle Object Storage Adapter**: Reads/writes objects (files) to Oracle Cloud Infrastructure (OCI) Object Storage.
- **Amazon S3 Adapter**: Enables integration with Amazon S3 buckets for file operations.

Messaging Adapters: Used for asynchronous communication and event-driven architectures.

- **Oracle Messaging Cloud Adapter**: Connects to Oracle Messaging Cloud Service for pub-sub scenarios.
- **JMS Adapter**: Integrates with Java Message Service providers to send/receive messages on queues and topics.
- **Kafka Adapter**: Connects with Apache Kafka for stream processing and event consumption.

Integration Types

In Oracle Integration Cloud (OIC), several integration methods (also called integration styles or patterns) are available to suit different business and technical needs. These styles determine how data flows between systems and how integration logic is executed.

App-Driven Orchestration: An event-driven integration that is triggered when an event occurs in a source system (e.g., creation of a record). It typically involves request-response or synchronous flows.

Example:

When a new customer is created in Salesforce, it triggers an integration that creates a corresponding customer record in Oracle ERP Cloud.

- **Trigger**: Salesforce sends a message when a new customer is created.
- **Flow:** The integration uses Salesforce Adapter (trigger) and Oracle ERP Adapter.
- **Response**: Sends back success/failure confirmation to Salesforce.

Scheduled Orchestration: This integration runs at defined time intervals (e.g., hourly, daily). It's suitable for batch processing where large data sets are transferred.

Example:

Every night at 12 AM, product inventory data from an on-premise database is uploaded to an Oracle SCM Cloud system.

- **Trigger:** Scheduler (time-based, no event).
- **Flow:** Reads data using DB Adapter via Connectivity Agent → Transforms it → Sends to Oracle SCM Cloud.
- **Result:** Inventory updates synced daily.

Basic Routing (Point-to-Point Integration): A lightweight integration with minimal transformation. It simply routes data from source to target systems with basic mapping.

Example:

A CSV file received via FTP is directly uploaded to a document management system like Oracle Content & Experience Cloud.

- **Trigger:** FTP Adapter reads the file.
- **Flow**: Directly passes it to content storage with basic metadata mapping.
- **Use Case Suitability**: Suitable for simple file transfers or message forwarding.

Publish and Subscribe (Pub-Sub Model): In this asynchronous and decoupled pattern, a publisher sends a message/event, and multiple subscribers consume and process the message independently.

Example:

A purchase order is created in Oracle ERP Cloud, and the event is published. Multiple systems—Inventory, Finance, and Logistics—subscribe to this event to take respective actions.

- **Publisher Integration**: Publishes PO creation event to a topic.
- **Subscriber Integrations**: Listen to the topic and execute different flows (e.g., updating stock, creating invoices, scheduling shipments).

Connections in OIC

Connections represent a defined setup used to link Oracle Integration Cloud with external or internal systems. They hold the essential configuration for communication, including security credentials, endpoint URLs, and specific adapter properties. Each connection is built using an adapter and can be reused across multiple integrations.

Types of Connections in OIC:

REST Connection:

- Used to connect to RESTful web services.
- Configure endpoint URL, HTTP methods, headers, query parameters, and authentication.
- Example: Integrate OIC with a third-party API like Google Maps or a payment gateway.

SOAP Connection:

- Connects to SOAP-based web services using WSDL.
- Used in legacy system integrations that expose SOAP interfaces.
- Example: Connect OIC to an insurance system that uses SOAP-based APIs.

FTP Connection:

- Enables file transfer operations (read/write) from/to remote servers.
- Secure credentials, host, port, and file path are required.
- Example: Daily upload of sales files from a retail POS system to Oracle Cloud.

File Connection:

- Used for accessing local or shared file systems (with agent support).
- Example: Integrating with on-premise systems to pick up HR documents from a network folder.

Database (DB) Connection:

- Connects to on-premises or cloud databases.
- JDBC URL, username/password, and agent setup (if on-prem) are needed.
- Example: Sync customer records from an Oracle DB to Salesforce.

Messaging Service Connection:

- Connects to services like Kafka, JMS, or Oracle Messaging Cloud.
- Used for asynchronous message publishing/subscription.
- Example: Consuming customer events from a JMS queue.

Social Media & IoT Connections:

- Specialized for platforms like Twitter, Facebook, and IoT frameworks.
- Used in scenarios requiring real-time monitoring or social engagement.
- Example: Capturing tweets mentioning a brand and storing them in a CRM.

Monitoring

Oracle Integration Cloud offers robust and comprehensive monitoring features that enable users to track the performance, status, and errors of integrations in real time. Monitoring helps ensure that integrations run as expected, exceptions are handled efficiently, and stakeholders are informed of critical events.

Instance Tracking: Instance Tracking provides detailed logs and records of every integration instance that is executed. It includes:

- **Execution Time**: Start time, end time, and duration of the integration.
- **Status**: Whether the integration succeeded, failed, or is in progress.
- **Payloads**: Optionally logs incoming and outgoing payloads for debugging.
- **Activity Logs**: Shows each step of the integration and corresponding results.
- **Search and Filter**: Allows filtering by status, integration name, business identifiers, or date range.

Example: A finance integration triggered by an invoice submission shows an instance record with status "Success," duration of 2.5 seconds, and steps like REST call, mapping, and DB write.

Business Identifiers: Business Identifiers allow tracking and correlation of integration instances based on meaningful business data. Instead of searching by technical instance ID, users can search by fields like:

- **Order ID**
- **Customer Number**
- **Invoice Number**

Example: A sales integration includes "CustomerID" as a business identifier so users can view all integration instances related to a specific customer.

Alerts: OIC supports setting up automated alerts that notify administrators or business users of critical issues:

Types of Alerts:

- Integration Failures
- Long-running Instances
- Missing Payloads
- Agent Communication Failures

Configuration:

- Set thresholds (e.g., failure after 3 retries)
- Define recipient lists and escalation procedures.

Example: An alert is configured to send an email to the integration support team if the customer onboarding integration fails more than twice in 10 minutes.

Configure Invoke Connection and Create an Integration

Configuring an invoke connection in Oracle Integration Cloud (OIC) involves setting up a connection to a target application that will be called during an integration flow. This is a critical step in application integration where one system sends data or requests actions from another system.

Step 1: Create a Connection for the Target Application

- Navigate to the Connections page in OIC.
- Click on Create and choose the appropriate adapter (e.g., REST, SOAP, Oracle ERP Cloud, Database, etc.).
- Provide the connection name and description.

- Configure the connection details like base URL, credentials, security policy, and any required headers or WSDL (for SOAP).
- Test the connection to ensure it's successfully configured.

Step 2: Use the Connection as an "Invoke" in Integration

- In your integration flow, drag the Invoke action from the palette.
- Select the connection you created as the endpoint.
- Choose the specific operation or service you want to call (e.g., POST method of a REST API or a stored procedure in a DB).
- This action will act as the outbound call to the external application.

Step 3: Define the Operation and Map Input/Output

- After choosing the operation, OIC will generate request and response schemas.
- Use the Mapper tool to map incoming data from the trigger/source to the fields required by the invoke target.
- Similarly, map the response data to an integration variable or the response structure.
- Apply transformation logic if needed, using functions, expressions, and lookups.

Step 4: Add Error Handling and Tracking

- Use a Scope to group actions and attach a Fault Handler for graceful error recovery.
- Log error messages and integration state using Logger actions.
- Enable tracking fields and business identifiers to facilitate monitoring.
- Optionally, configure retry logic, timeouts, and alert notifications in case of failure.

3.3.3 Oracle cloud Infrastructure Data integration

Oracle Cloud Infrastructure Data Integration (OCI DI) is a fully managed, serverless, cloud native ETL (Extract, Transform, Load) service that helps developers, data engineers, and business users efficiently move, transform, and load data from various sources into data lakes, data warehouses, and analytics platforms. Built on Oracle’s next-gen infrastructure, OCI DI enables seamless integration with other OCI services, enhancing agility and scalability. It caters to a wide range of data engineering and business intelligence needs by integrating deeply with

other OCI services like:

- Oracle Object Storage
- Autonomous Database (ADB/ADW)
- Oracle Analytics Cloud (OAC)
- Oracle Data Flow (Apache Spark-based processing)

Oracle Data Integration (OCI DI) provides a user-friendly and structured approach to designing and managing data workflows by organizing its core functionality into six foundational components that together streamline the ETL development lifecycle.
At the highest level, a Project acts as a logical workspace or container, allowing users to group together related assets such as data flows, pipelines, tasks, and configurations.

This enables efficient management, version control, and collaboration among teams working on the same data integration initiative. Within a project, the Data Flow serves as the heart of the ETL process. It is a visual representation of the logic that governs how data is extracted from sources, transformed through a series of operators (such as joins, filters, aggregations, expressions), and finally loaded into the target systems. The drag-and-drop interface for data flows significantly simplifies the creation of complex transformations, making it accessible even to non-developers. To manage the execution of one or more data flows or other tasks, OCI DI uses a Pipeline, which acts as an orchestration layer. A pipeline defines the order of execution, error-handling logic, parallelism, and dependencies between different tasks, allowing users to build scalable and modular ETL workflows that can adapt to business logic and scheduling needs.

A Task in this context is the most granular executable component. It can be a Data Flow Task that runs a defined data flow, a SQL Task that runs SQL queries against connected databases, or other custom executions. Tasks are the building blocks that pipelines link together to form end-to-end workflows. For simpler or smaller use cases, OCI DI offers the Data Loader, which is a streamlined utility designed for rapid ingestion of flat files like CSVs or Excel sheets into target systems. This component is especially useful for quick, one-off data loading jobs or for users who need minimal transformation capabilities. Finally, an Application in OCI DI groups one or more pipelines and supporting artifacts into a deployable and schedulable unit. Applications are used to manage and deploy end-to-end solutions that run on defined triggers or schedules, enabling repeatable and automated data integration across various environments.

Product Features

Few of the product features are listed below:

- **Visual Flow Designer:** OCI DI provides a no-code/low-code visual interface known as the Visual Flow Designer, enabling users to build data transformation workflows using drag-and-drop components. This simplifies pipeline creation, especially for users with limited coding experience.
- **Out-of-the-Box Operators:** A rich set of prebuilt transformation operators—such as Join, Filter, Union, Aggregate, Lookup, Normalize, Pivot/Unpivot, and Expression Transform—are available to handle complex data manipulation with ease and speed.
- **Transformation Expressions Engine:** OCI DI supports a SQL-like expression editor that allows users to define custom transformation logic within data flows. This enables highly granular control over how data is processed, calculated, and transformed.
- **Runtime Monitoring and Observability:** The platform includes a robust runtime monitoring system that tracks pipeline execution status, logs, and performance metrics. It also provides visual lineage views to understand the flow and dependencies of data transformations.
- **Connectivity to Diverse Data Sources:** OCI DI offers native connectivity to Oracle Cloud services such as Autonomous Database, Object Storage, and Oracle Analytics Cloud. Additionally, it supports JDBC-based connectors for third-party platforms like Snowflake, Microsoft SQL Server, BigQuery, and others.
- **Parameterized Pipelines:** With runtime parameter support, OCI DI pipelines can be designed as reusable templates. Users can configure different input/output sources or transformation logic dynamically, enhancing modularity and flexibility.
- **Enterprise-Grade Security Integration:** Security is tightly integrated through OCI IAM for access control, OCI Vault for key management, and Private Endpoints for secure VCN-based communication. All data is encrypted both at rest and in transit, ensuring enterprise compliance.

Core Concepts

Workspace: A Workspace in OCI DI serves as an isolated development environment where users can create, manage, and test all their data integration artifacts. It acts as a boundary for collaboration, allowing multiple users or teams to work on their respective projects

independently without interfering with one another. Each workspace maintains its own set of projects, data flows, pipelines, and configurations, ensuring modular and secure development.

Data Asset: A Data Asset defines the connection details to a specific data source or target system. It includes metadata such as the connection string, authentication method, and driver type (e.g., Oracle DB, MySQL, Object Storage, Snowflake). Data assets are reusable components and act as foundational connectors that enable data flows and pipelines to interact with external systems securely and efficiently.

Application: An Application is a deployable package that bundles together one or more pipelines and their associated tasks. It provides a higher-level abstraction for managing and scheduling complete integration solutions. Once developed, an application can be deployed, scheduled, and monitored as a single unit, facilitating operational consistency and ease of deployment across environments.

Task Scheduler: The Task Scheduler is responsible for managing the execution of data integration workflows based on pre-defined schedules or triggering events. Users can configure time-based triggers (e.g., hourly, daily, weekly) or event-based triggers (e.g., when a new file lands in Object Storage or when a database update occurs). This enables automated and timely execution of pipelines without manual intervention.

Execution Context: The Execution Context defines the runtime environment where data integration processes are executed. It specifies the compute shape, memory allocation, and parallelism settings that govern how data flows and tasks are processed. Choosing the appropriate execution context ensures optimal performance and resource utilization based on the complexity and volume of the data being handled.

Key Capabilities

The Key Capabilities of Oracle Cloud Infrastructure Data Integration (OCI DI) refer to the fundamental strengths and features that make the platform efficient, scalable, secure, and user-friendly for modern data integration needs. These capabilities define how well the tool can support building, managing, and automating ETL/ELT workflows in a cloud-native environment. Few of them are listed below:

- **Serverless Architecture:** OCI DI is built on a serverless architecture, meaning users do not need to provision, configure, or manage any infrastructure. Oracle automatically handles the compute resource allocation, autoscaling, and workload distribution in the background. This simplifies deployment, reduces operational overhead, and allows teams to focus entirely on designing data logic.

- **Declarative and Visual Design:** The platform supports both visual and declarative design paradigms. Users can create ETL workflows using an intuitive drag-and-drop interface or define the same logic programmatically using REST APIs or SDKs. The declarative approach ensures that transformation logic is clearly documented and easier to maintain, while the visual designer promotes faster development and onboarding for less technical users.

- **Modularity and Reusability:** OCI DI promotes modular design principles by enabling the reuse of key components such as transformation expressions, joins, data mappings, connections, and parameters across multiple data flows and pipelines. This not only accelerates development but also improves consistency and maintainability of integration projects over time.

- **Secure Connectivity:** Security is deeply integrated into OCI DI through multiple layers. It leverages OCI Vault for secure storage and management of authentication credentials and encryption keys. Private Endpoints allow the service to connect to private data sources within a Virtual Cloud Network (VCN) without exposing them to the public internet.

- **Operational Intelligence:** The platform offers comprehensive operational visibility through dashboards that display job execution history, real-time monitoring, and performance statistics. Users can drill down into task-level logs, trace errors, and view data lineage to diagnose issues effectively. This level of insight is crucial for ensuring pipeline reliability & tuning performance,

OCI Data Integration (OCI DI) v/s Oracle Data Integrator Marketplace (ODI_MP)

Table 3.4 Difference between OCI DI & ODI MP

Criteria	OCI Data Integration (OCI DI)	Oracle Data Integrator (ODI_MP)
Deployment	Fully managed, no provisioning	VM-based, self-managed via OCI Marketplace
Architecture	Serverless, elastic	Agent-based, monolithic
UI/UX	Web-based visual design tool	Java-based ODI Studio Desktop
Execution Engine	Native cloud runtime	Standalone/Java EE ODI Agent
Ease of Use	Simplified, low-code/visual	More complex; enterprise-grade transformations
Use Case Focus	Modern cloud-first ETL/ELT	Legacy system integrations and hybrid workloads
Connector Support	Oracle Cloud + JDBC sources	Wide enterprise integration support including SAP, mainframes
Scalability	Auto-scaled	Manual provisioning
Security Management	IAM, VCN, Vault	Requires additional security configuration
Monitoring and Ops	Integrated with OCI monitoring tools	Requires setup of ODI Console and agents

With reference to above table 3.4 OCI DI is a fully managed, serverless, and cloud-native data integration service with a web-based, low-code interface, ideal for modern ETL/ELT workflows in Oracle Cloud. It offers auto-scaling, simplified operations, and built-in security and monitoring using OCI services.

In contrast, ODI_MP is a VM-based, self-managed solution designed for enterprise-grade data transformations, with a Java-based desktop interface. It supports a broader range of legacy, but requires manual provisioning, security and monitoring, making it more complex to manage.

3.4 Final Phase

3.4.1 Oracle Apex

Introduction to Oracle Application Express

Oracle Application Express (APEX) is a web-based, low-code development platform that enables users to build data-driven applications quickly and with minimal programming. It is a fully supported feature of the Oracle Database and is ideal for creating secure, scalable enterprise applications. Since APEX runs entirely in a web browser, developers do not need to install any additional software. It is used by both beginners and professionals for rapid development of data-centric applications.

What is Oracle Apex?

Oracle Application Express (APEX) is a web-based, low-code application development platform that allows users to build secure, scalable, and data-driven applications with minimal effort. Built directly into Oracle Database, APEX eliminates the need for complex installation or third-party tools—developers only need a web browser to begin application development.

- **Low-Code Platform:** APEX uses a visual development interface, drag-and-drop components, and wizards, enabling even non-programmers to create apps. Developers can focus more on logic and data flow rather than syntax.

  Example:

  A business analyst without programming experience can use APEX to create a sales dashboard that fetches data from Oracle tables and displays key performance indicators (KPIs) like revenue, top-selling products, and trends—all without writing much code.

- **Tightly Integrated with Oracle Database:** Since APEX is part of Oracle Database, it can directly access and manipulate data using PL/SQL and SQL.

  Example:

  A logistics company stores all shipment data in an Oracle Database. With APEX, they create an internal web application to track shipment status, delays, and delivery performance across regions using just SQL and built-in charts.

- **Runs Entirely in a Web Browser:** Users only need a browser to develop and use APEX applications—no special software or plugins are required. This makes it platform-independent and accessible from anywhere.

  Example:

  A university creates an online student registration portal using APEX. Students and

administrators can access it through a browser to enroll in courses, submit documents, and generate grade reports—no software installation needed.

- **Used by Beginners and Professionals:** The low-code environment is beginner-friendly, while experienced developers can use advanced features like REST APIs, dynamic actions, JavaScript integration, and PL/SQL procedures.

  Example:

  A student builds a basic library management system as part of a college project. Meanwhile, an IT department at a large bank develops a complex APEX-based loan approval workflow system that integrates with external APIs, automates approvals, and handles user authentication via Oracle SSO.

- **Ideal for Rapid Prototyping and Agile Development:** APEX is widely used in scenarios where speed and flexibility are critical prototypes can be built in hours and refined iteratively based on feedback.

  Example:

  A startup uses APEX to build a customer feedback app within a day to collect reviews from their product launch event. They continue to enhance it based on real-time user input.

Benefits

Benefits include a low-code approach to application development, significantly reducing time and effort. It allows integration with RESTful web services, data visualization, user authentication, and security mechanisms.

- **Low-Code Development:** APEX offers a graphical interface to rapidly build forms, reports, dashboards, and more.
- **Rapid Prototyping & Deployment:** Developers can quickly create working prototypes and deploy them seamlessly in real-time.
- **RESTful Web Services Integration:** Can easily consume or publish REST APIs for integrating with external systems.
- **Data Visualization:** Built-in charting, interactive reports, and dashboards using libraries like Oracle JET.
- **User Authentication & Security:** Includes session management, user roles, access control, and encryption for securing data.

- **Reusable Components:** Templates, plug-ins, prebuilt page types, and themes boost consistency and speed.

Architecture

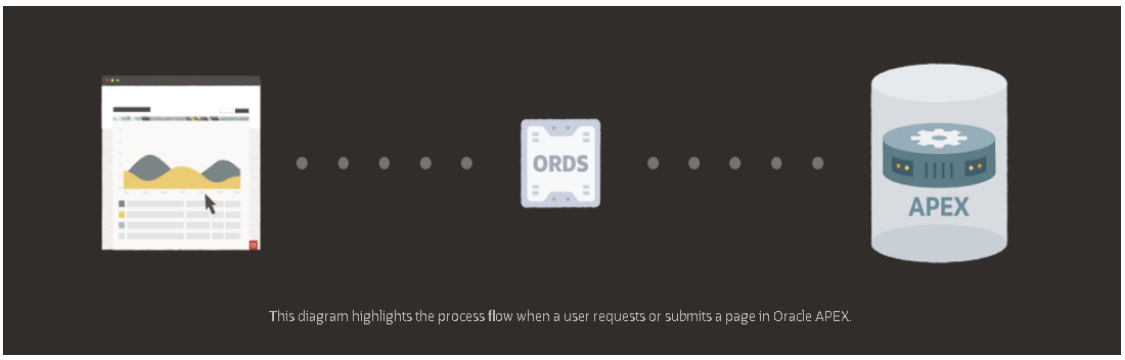

Figure 3.7 Architecture of Oracle Apex

From above figure 3.7, Oracle APEX is a web-based application development platform that follows a three-tier architecture:

1. Client Tier (Web Browser)

- The user interacts with the APEX application via a web browser.
- All requests (e.g., button clicks, form submissions) are sent to the web server.

2. Middle Tier (Oracle REST Data Services - ORDS)

- ORDS serves as the bridge between the browser and the Oracle Database.
- It interprets HTTP requests from the browser and converts them into database calls.
- Also serves static resources like CSS, JS, and APEX components.

3. Database Tier (Oracle Database with APEX)

- The core of APEX resides in the Oracle Database schema.
- Business logic, data processing, and rendering decisions are handled using PL/SQL.
- Returns HTML pages or JSON back to ORDS, which sends it to the browser.

Major Components of Oracle Apex

Oracle APEX (Application Express) offers a comprehensive suite of tools that streamline application development, database management, team collaboration, and integration with external services. Here's an in-depth look at its four major components:

1. **SQL Workshop**: SQL Workshop is a robust environment within Oracle APEX designed for database developers and administrators. Key features include:
   - Object Browser: Provides a tree-view interface to explore and manage database objects like tables, views, indexes, and sequences.

- SQL Commands: A web-based SQL editor that allows for the execution of ad-hoc SQL statements, facilitating quick data retrieval and manipulation.
- SQL Scripts: Enables users to upload, store, and execute SQL scripts, promoting reusability and efficient script management.
- Query Builder: Offers a graphical interface to construct complex SQL queries using drag-and-drop functionality, simplifying the query creation process.
- Data Workshop: Assists in loading and unloading data in various formats, such as CSV or Excel, into the database, streamlining data import/export tasks.

2. App Builder: App Builder serves as the central hub for developing applications within Oracle APEX. Its capabilities include:

- Create Application Wizard: Guides users through a step-by-step process to build applications, allowing for the selection of themes, pages, and features.
- Page Designer: A powerful interface that enables developers to design and customize pages, manage components, and define behaviors interactively.
- Component View: Provides a hierarchical view of application components, facilitating easy navigation and management.
- Shared Components: Allows for the reuse of elements like templates, lists of values, and navigation menus across multiple applications, ensuring consistency.
- Responsive Design: Ensures that applications are mobile-friendly and adapt seamlessly to various screen sizes.

3. Team Development: Team Development is a collaborative toolset within Oracle APEX that supports the entire application lifecycle. Its features include:

- Features Tracking: Allows teams to document and monitor new functionalities from conception to implementation.
- To-Dos: Enables the assignment and tracking of tasks that are not directly tied to specific features, ensuring all development activities are accounted for.
- Bugs Management: Provides a structured approach to logging, assigning, and resolving defects, enhancing application quality.
- Milestones: Helps in planning and tracking significant project events, aligning development efforts with project timelines.
- Feedback: Facilitates the collection of user input directly within the application, promoting continuous improvement based on real user experiences.

By integrating these tools, Team Development fosters effective collaboration, transparency, and accountability within development teams.

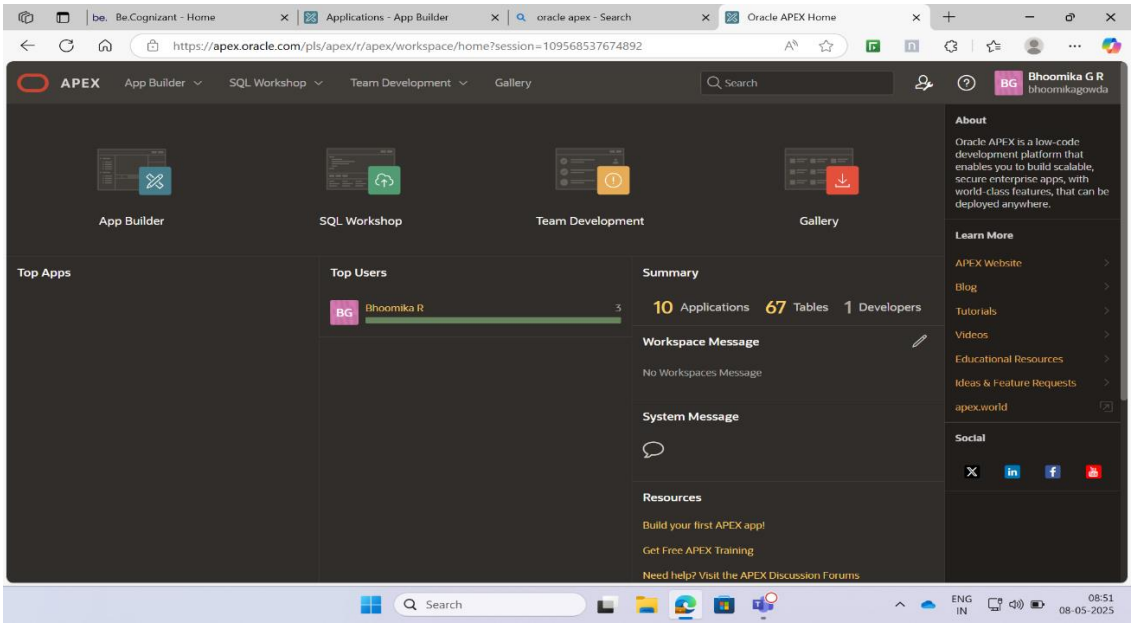

Figure 3.8 Homepage Oracle Apex

The above figure 3.8 shows Oracle APEX Home Dashboard is the central landing page after logging into a workspace. It provides quick access to key development tools like App Builder, SQL Workshop, Team Development, and Gallery. Users can view recent activity, workspace usage, and access learning resources from the right sidebar. This interface simplifies navigation and serves as the starting point for managing applications and database development.

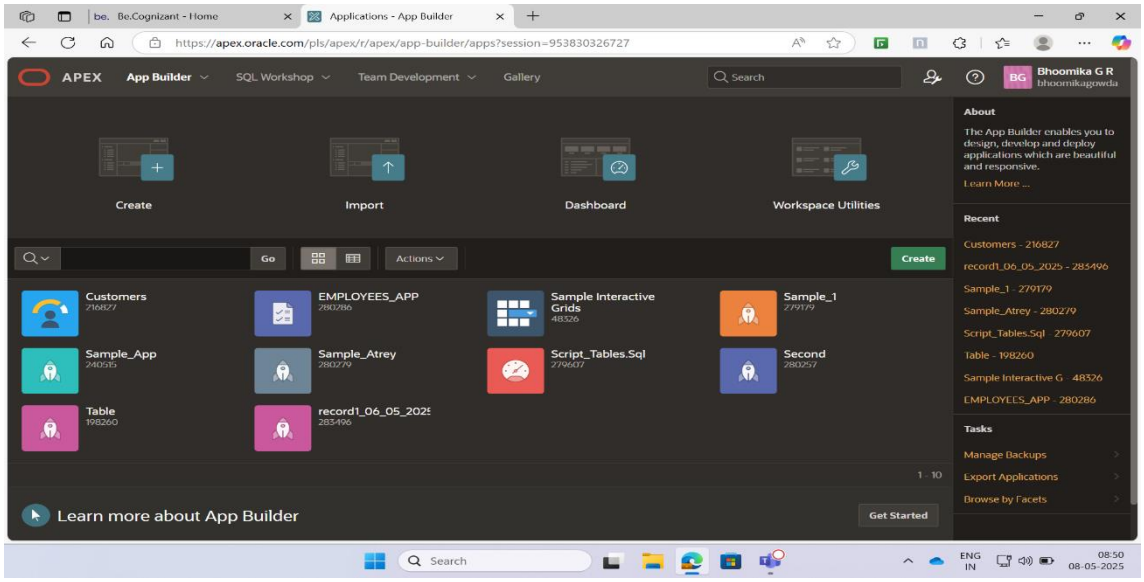

Figure 3.9 App Builder in Oracle Apex

The App Builder interface in figure 3.9 is the primary environment for creating, editing, and managing applications. It displays a list of available apps with their statuses and provides options to build new apps or import existing ones. With intuitive navigation and integrated development tools, this section enables developers to efficiently design responsive, data-driven applications using a low-code approach.

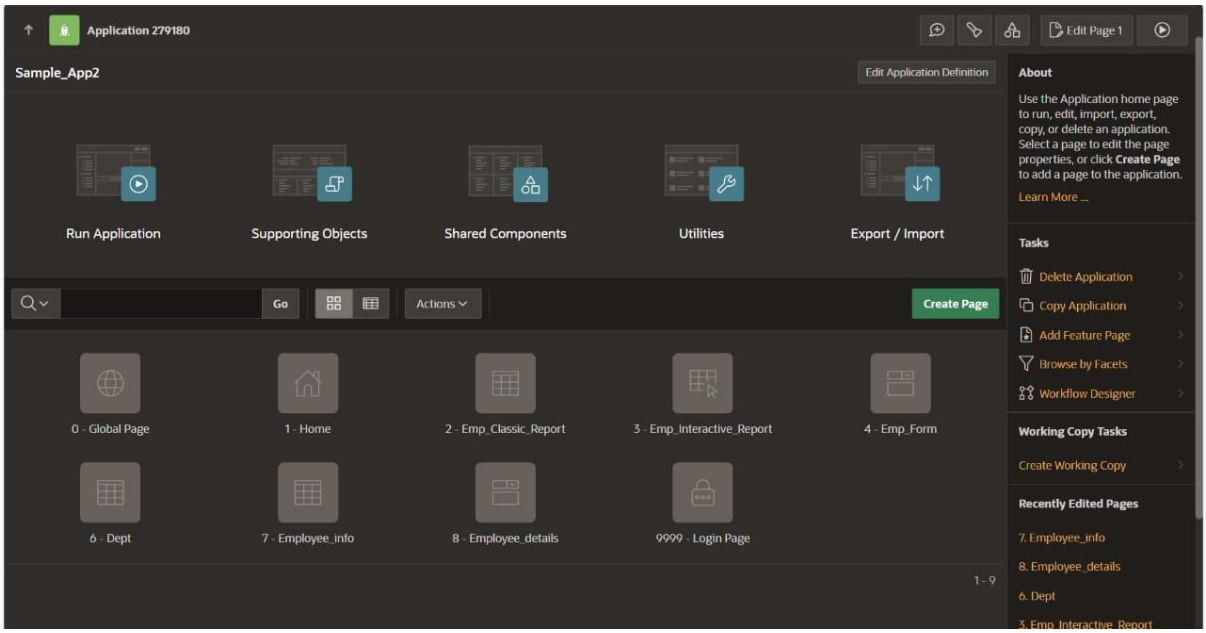

Figure 3.10 Application and pages Oracle Apex

The above figure 3.10 showcases the internal structure of a specific application within the Oracle APEX App Builder. It displays key tools such as Run Application**,** Supporting Objects**,** Shared Components**,** Utilities**,** and Export/Import, all of which help in managing the app's functionality and configuration. Below, the page layout presents individual components like Global Page**,** Home**,** Cust**,** and Login Page, indicating the application's page flow.

The below figure 3.11 represents the Page Designer interface of Oracle APEX, which is a powerful visual editor used to design and configure individual pages of an application.

The screen is divided into three primary sections:

1. Left Panel – Page Rendering Tree

This section shows the hierarchical structure of the page with components categorized into Pre-Rendering, Rendering (Body), and Post-Rendering. Developers can expand and manage regions, items, and layout positions from this tree. In this case, the page "Cust" is selected, showing detailed elements under the Body.

2. Center Panel – Layout Editor

The middle pane provides a visual representation of the page layout. It allows developers to

drag and drop components into layout positions like the banner, breadcrumb bar, body, and footer. The selected region named "Cust" is shown with its sub-regions like Edit, Delete, Previous, Next, etc., allowing for intuitive design control.

3. Right Panel – Property Editor

This panel displays configuration properties for the currently selected element, such as the name, template, CSS classes, and navigation settings. It also includes JavaScript integration points and custom behavior settings, making it easy to fine-tune each element's appearance and logic.

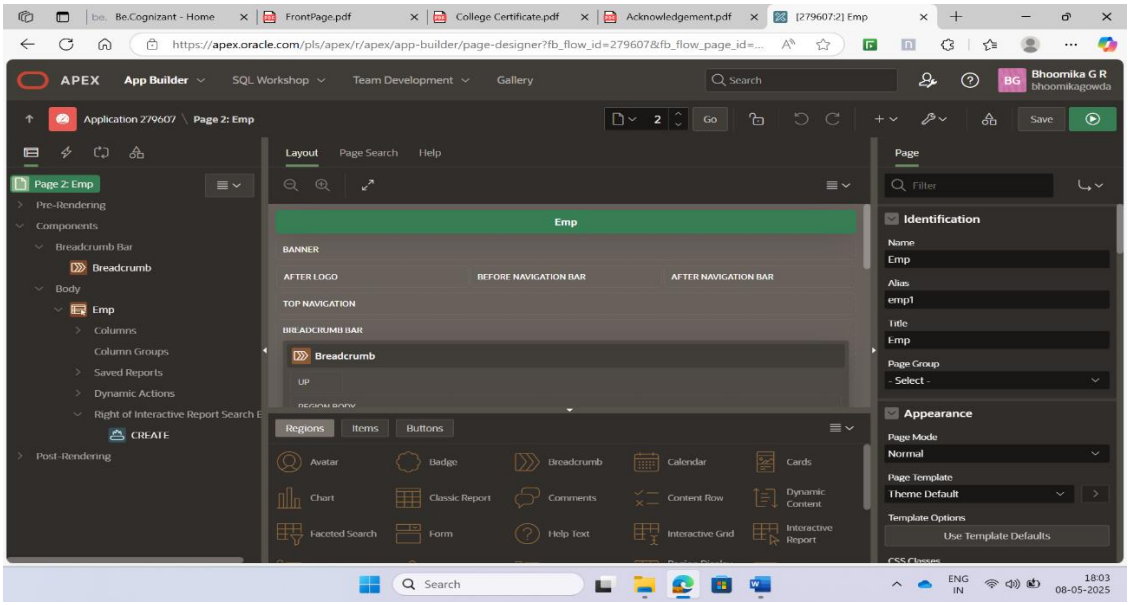

Fig 3.11 Page designer View Oracle Apex

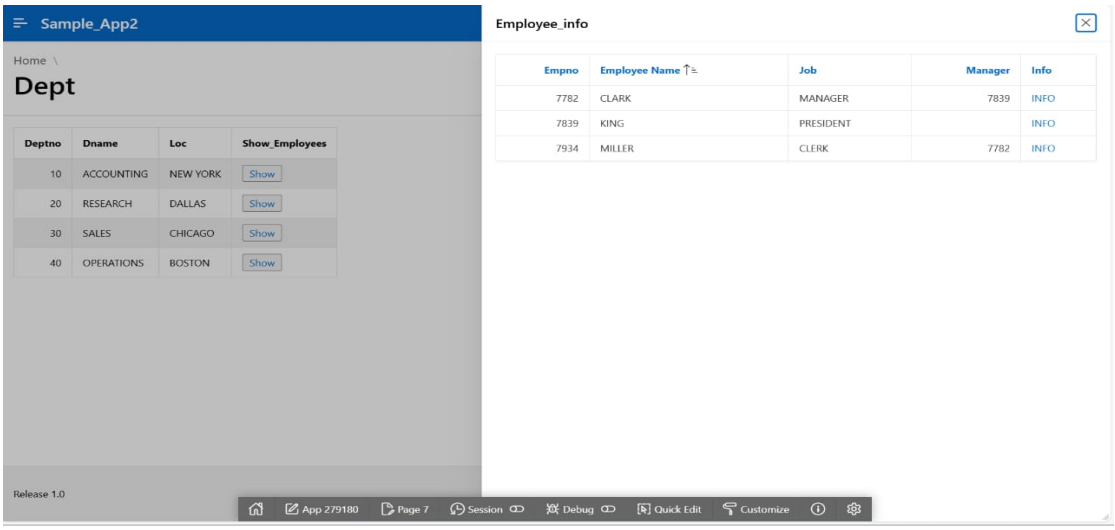

Fig 3.12 Webpage displaying different tables.

The above figure 3.12 shows a table listing customer interactions (e.g., meetings, calls) with audit details like who created/updated each record and when. Simple interactive report layout and displays a department list with a "Show" button for each row that opens a modal dialog showing employees in that department. Demonstrates a master-detail relationship using a popup.