Name: Naga Venkata Kanakalakshmi Murikipudi
UNT ID: 11725119
Project: part 1

Group Number: 8
Group Members:

Name	UNT ID	Mail ID
Naga Venkata kanakalakshmi	11725119	nagavenkatakanakalmurikipudi@my.unt.edu
Vishnu Vardhan Reddy Sudireddy	11642773	vishnusudireddy@my.unt.edu
Srinivas Sankula	11667743	srinivassankula@my.unt.edu
Sai Sindhu Rudraraju	11644475	saisindhurudraraju@my.unt.edu

Introduction:
AutoZone, a national retailer of automotive parts and accessories Company has requirement to store the information of the inventory, employees, insurance, servicing, payment plans, supplier details and customer details. AutoZone also want to store the information about locations, jobs done by the employees and automotive retailer branch details.

Description:

- Each retailer in the AutoZone chain needs to be identified based on the retailer id. It can have the basic information like contact, business hour and each AutoZone retailer can have a manager and the website details related to the that location. Multiple employees can work in the one automotive retailer. Each retailer can have in house inventory and external inventory. Automotive retailer address/location needs to be stored. Each automotive retailer can afford multiple jobs or services.
- Inventory product entity serves as a major component and uniquely identified in this system. It manages the various automotive products. It includes the attributes like name, quantity, price, automotive_retailer_id, automobile_id, address_id. It holds the details of the automotive retailer, automobile data. It establishes the relation with suppliers, automotive retailers, part service, automobile, address and bill.It serves as a central component for efficient inventory management, procurement, and tracking within the system.
- Employee entity holds the data of the employees working in AutoZone. Each employee is uniquely identified with their ID. Along with the ID this entity will also have the attributes first name, last name, date of birth, phone number, email address, annual salary, ssn, address, hire date and the automotive retailer ID. An employee can create many bills for the customers and so the employee entity will form a one-to-many relation with the bill entity. More than one employee can have the same address as there is a chance of 2 employees living in the same location. So, employee will form a many to one relationship with the address entity. Many employees can be part of a job and one employee can be part of many jobs. So, employee entity will form many to many relationships with the job entity. As Many employees work in one location, employee entity will form a many to one relationship with the automotive retailer entity. Employees may receive multiple paychecks over a period. So, Employee entity will form a one-to-many relationship with employee payroll entity.
- Employee payroll entity has the record of paychecks given to employees. Each pay given to an employee is identified by a unique id. This entity has the attributes hours worked (number of

hours worked by employee during the pay cycle), start date (start date of the pay cycle), end date (end date of the pay cycle), pay (amount paid to the employee) and employee id (id of the employee that is used to uniquely identify an employee). The employee payroll entity will form a many to one relationship with the employee as one employee may receive many pays over a period.

- AutoZone can have suppliers to inventory who supply the products that stored or used during the services. An inventory can have multiple suppliers and vice versa. Suppliers' information like contact and address needs to be stored. Suppliers can uniquely identified by their id.
- Address entity serves as a global component within the system, defines location information. It is uniquely identified and includes the attributes like apartment number, street, city, state, country, and ZIP code, all of them are mandatory. This entity forms relationships with other entities in various ways: many employees may have a single address, inventory products are associated with specific addresses, automotive retailers and suppliers have distinct addresses, and multiple customers can be linked to one address.
- Bill entity is crucial for recording financial transactions. It is uniquely defined and includes attributes like date, payment mode, insurance, customer, job, employee, sale type (can be an online or offline), and payment plan. It forms relationships with various entities: multiple bills can be linked to one employee, have many-to-many connections with inventory products, and maintain multiple bills of each association with insurance, customers, and payment plans. Additionally, every job has a specific bill.
- Job entity is essential for managing automotive service tasks. It is uniquely defining and includes attributes like date, description, customers, automotive retailers, VIN numbers, and automobiles. All of which are mandatory. "Job" forms key relationships: It establishes a direct link with billing records, multiple jobs are associated with automobiles and automotive retailers, Customers. Various jobs can be done by different employees. Whereas Single job can be performed using multiple part services. This entity serves as a central hub for tracking and managing automotive service jobs.
- Part service like during the service which are automotive parts we are using to get the job done for a vehicle. It may have the information like job details, name/description of the service and it can have quantity of the parts which are using for that service. Part service can be identified by their id. Part service can have type like is it only service or any parts used to get that service done. In this many parts service can be related to one job and it have information related to the inventory products to know the parts related to the which inventory.
- Customer entity represents individuals in the system and is identified uniquely. It includes essential attributes like first name, last name, birthdate, driver's license, phone, and address. Multiple customers may opt for multiple insurance policies. A Customer can be associated with multiple jobs which has multiple bills. Many customers may have the single address. This entity enables efficient management of customer data, service history, and financial transactions.
- Insurance is like if customer wants to utilize his insurance plan for the payment, we need to store the information of that insurance. It can have policy type, provider and claim percentage. The particular insurance plan can be identified by plan id. A customer can have multiple insurance and vice versa. we would also like to include insurance id in the bill, it will be like one insurance plan can be included in many bills.
- Payment plan like if customer interested in the paying in instalments, he can use this option. This payment plan will have the information like plan name, number of instalments and the interest rate related to the payment plan. Each payment can be identified by plan id and multiple bills can have single payment plan.

- Would like to store the information about automobile. So that when the job is performed we can use the parts related to the specific automobile model. It can have the information like manufacture, name, variant, year of the build and color of the vehicle. Each automobile model can be identified by automobile id. An automobile can have multiple products in the inventory and also would like to keep the automobile information in the jobs performed. It will be like multiple jobs can be performed to an automobile.

Assumptions:

- A product can be available both in store and in warehouse. If the product has automative_retailer_id same as address id, that means the product is in store. And if automative_retailer_id is null, then the inventory is not in store.
- Automobile entity will have the generic information about the model, manufacturer and make of the automobile available in the market.
- A job will be created if a customer wants to get his automobile serviced in the store and the bill will be generated for the job.
- A bill can be generated without a job when a customer purchases products from the store without getting the service done in store where this data is stored in bill_inventory table.
- Employee can receive multiple payments over a period. This is maintained in employee_payroll table.
- Employee can be part of the job and he can generate bill for the job. And employee can sell the products to the customer who is not seeking service (not part of the job) from the retailer.
- Address is global table where the addresses of suppliers, customers, employees, inventory and retailer are stored.
- customer can pay the bill directly or he can opt for payment plan. Customer can select the payment plan form the payment_plan table.
- Job will be created if a customer opts services from the retailer. services are work done by the employees on the automobile to fix issues or install accessories.
- Part_services have the information about the products used in the job. One Job can have multiple products.
- Insurance table has different insurance policies that are available in the market.
- more than one person (customer or employee) can have same address. but no 2 suppliers and retailers can have the same address.
- product id in part service can be null as labor chargers of a job can also be clocked in part service where no inventory will be tagged to that service.
- The total amount of the bill of a customer who opted for a payment plan will be recorded as a transaction in bill table. And the installments are also saved as transactions in the same table.

Oracle Cloud Signup:

In this section we will discuss about oracle cloud installation and attached respective screenshots.

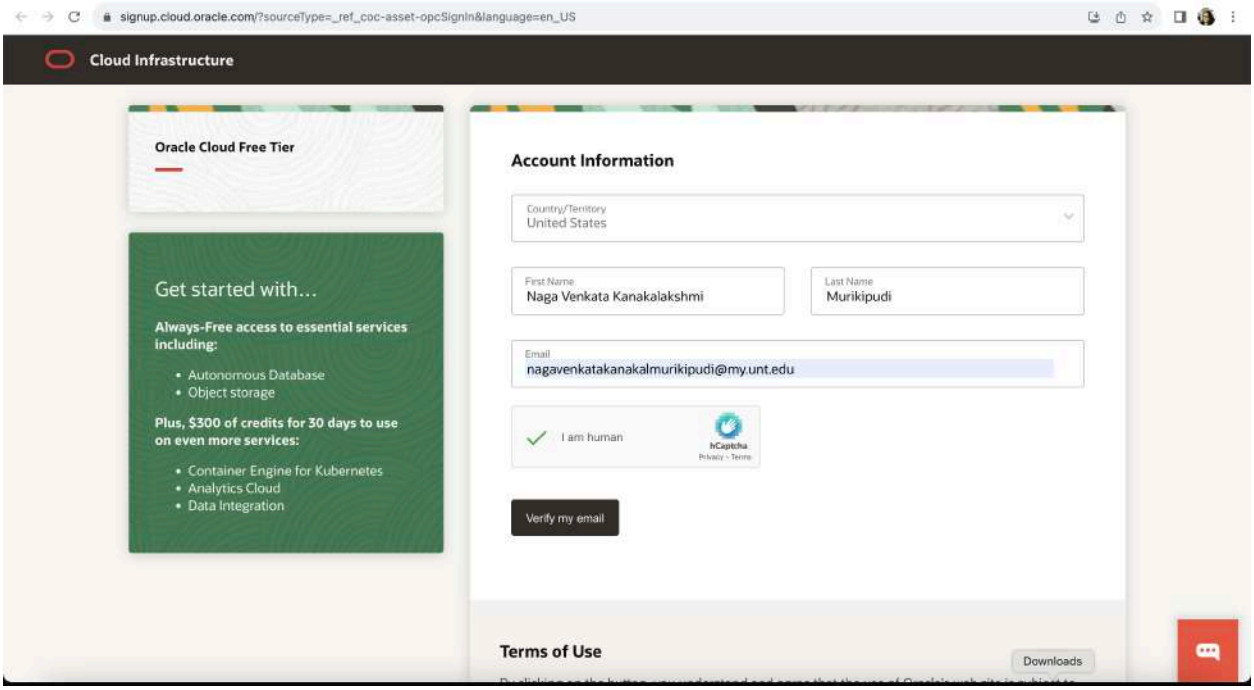

Figure 1: Screenshot of Account Information.

In Figure 1, I have given Account information that is Country, First Name, Last Name, Email. For Email verification clicked on “Verify my email”.

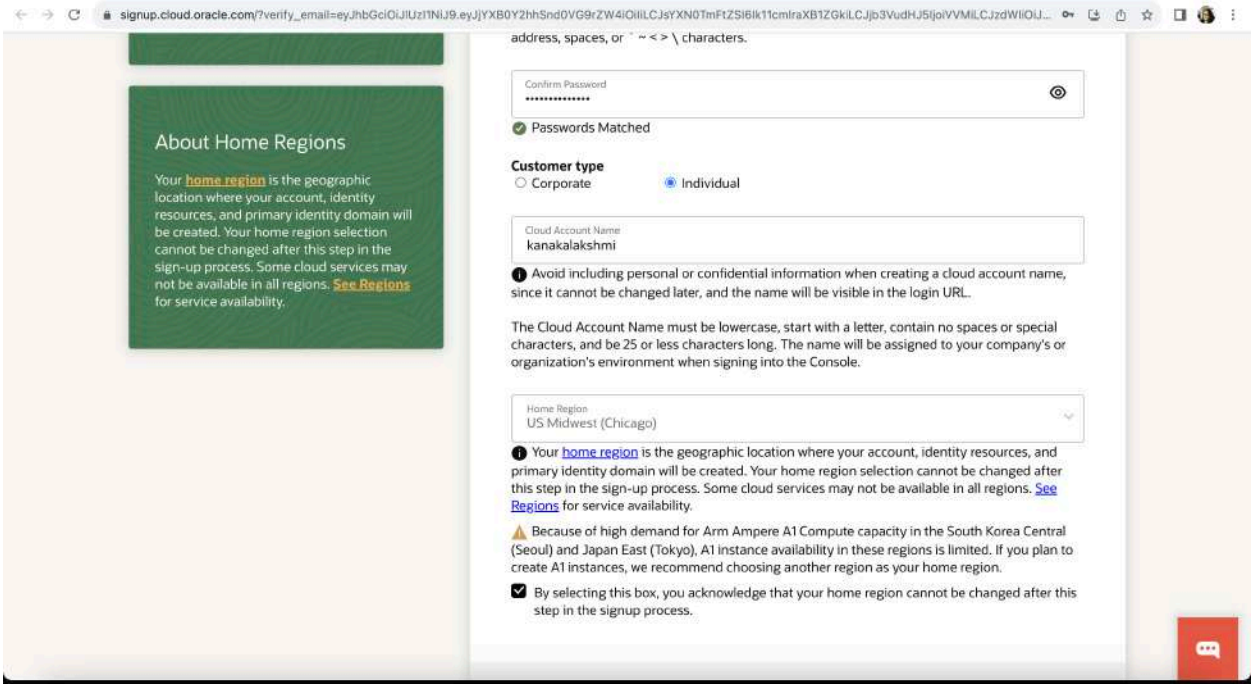

Figure 2: Screenshot of Cloud Account Name and Home Region.

Once after clicking on verify mail, which I received to my email, its redirected to Figure 2. Here I have given Password, Cloud Account Name and Home Region.

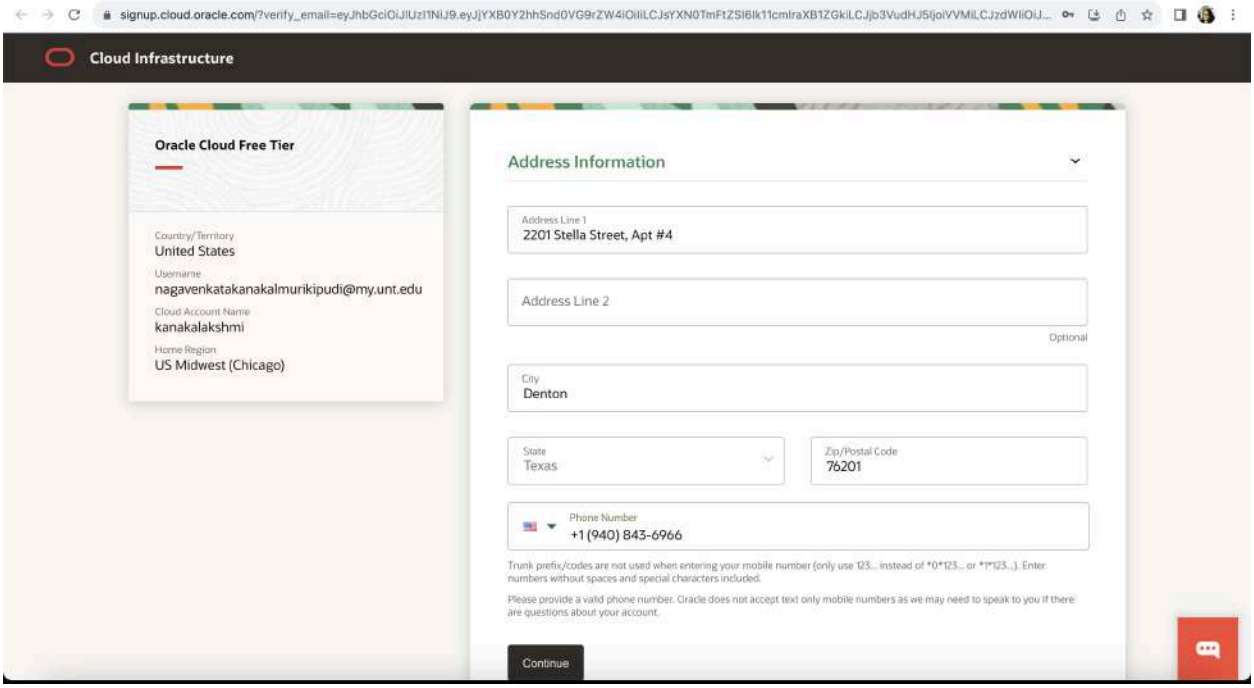

Figure 3: Screenshot of Address Information.

In Figure 3, I have given Address Information that is Address Line, City, State, Zip/Postal Code and Phone Number.

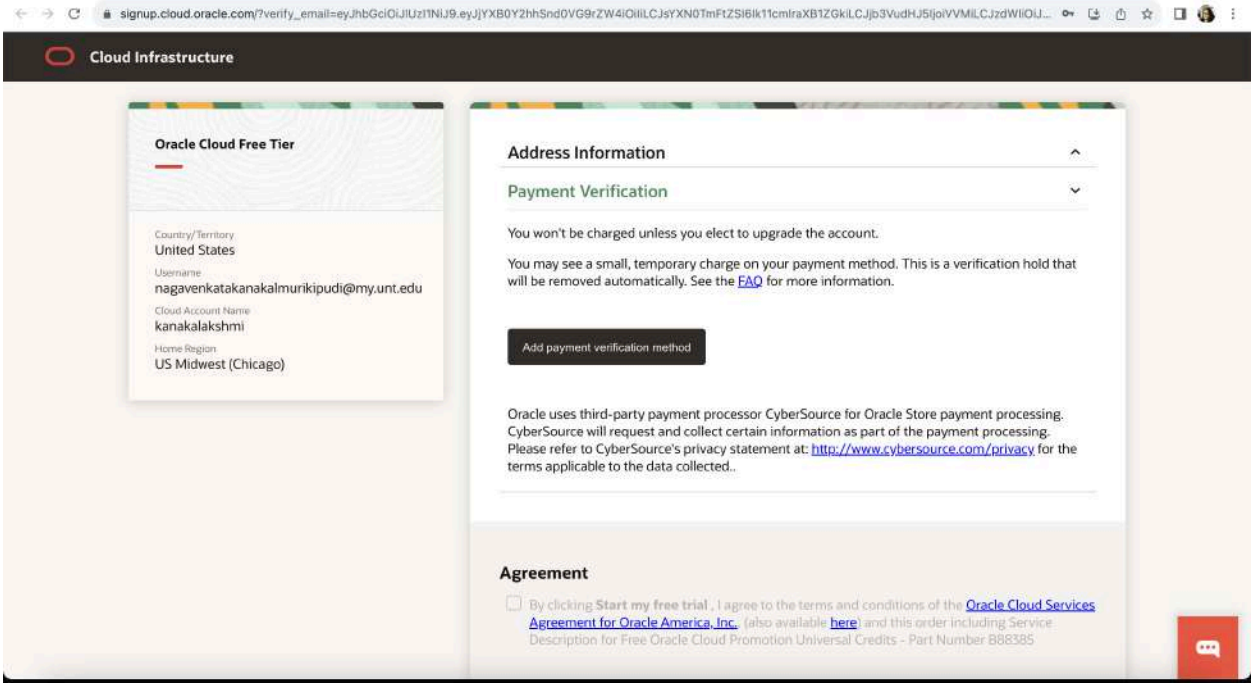

Figure 4: Screenshot of Payment Verification.

In Figure 4, I have gone through Payment Verification Details and Initiated Payment Process.

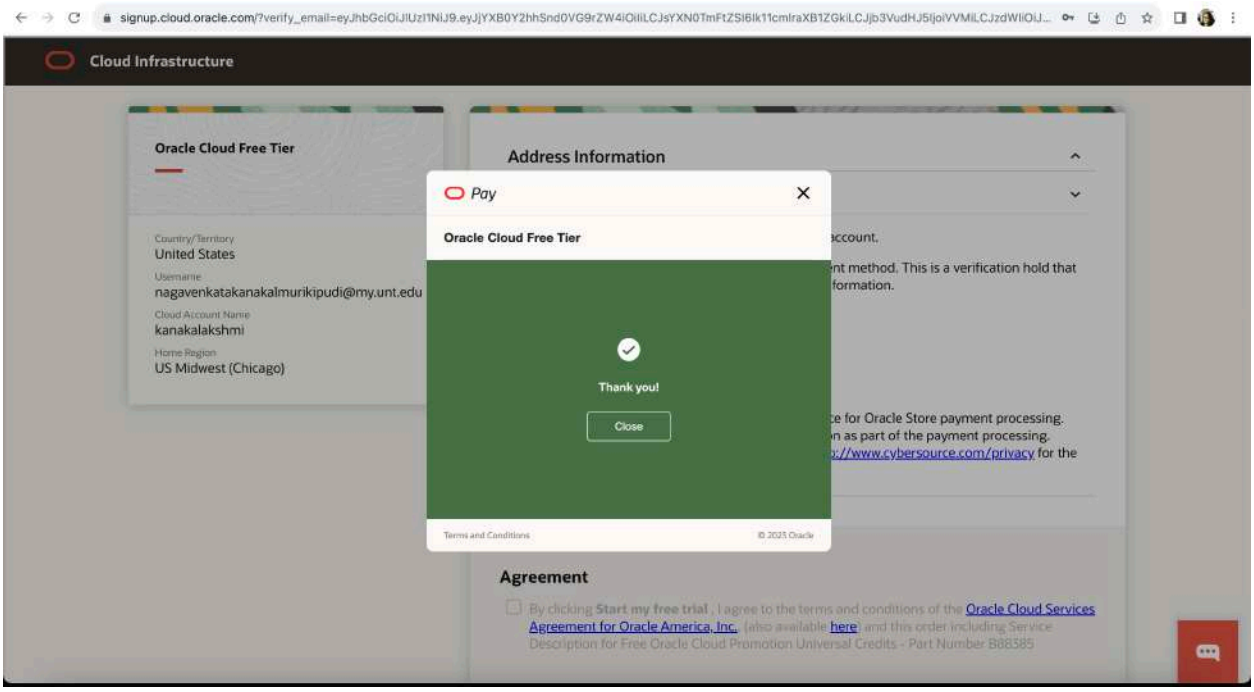

Figure 5: Screenshot of Payment Successful.

Figure 5 shows that payment is successfully done and got Thank you! Pop-up.

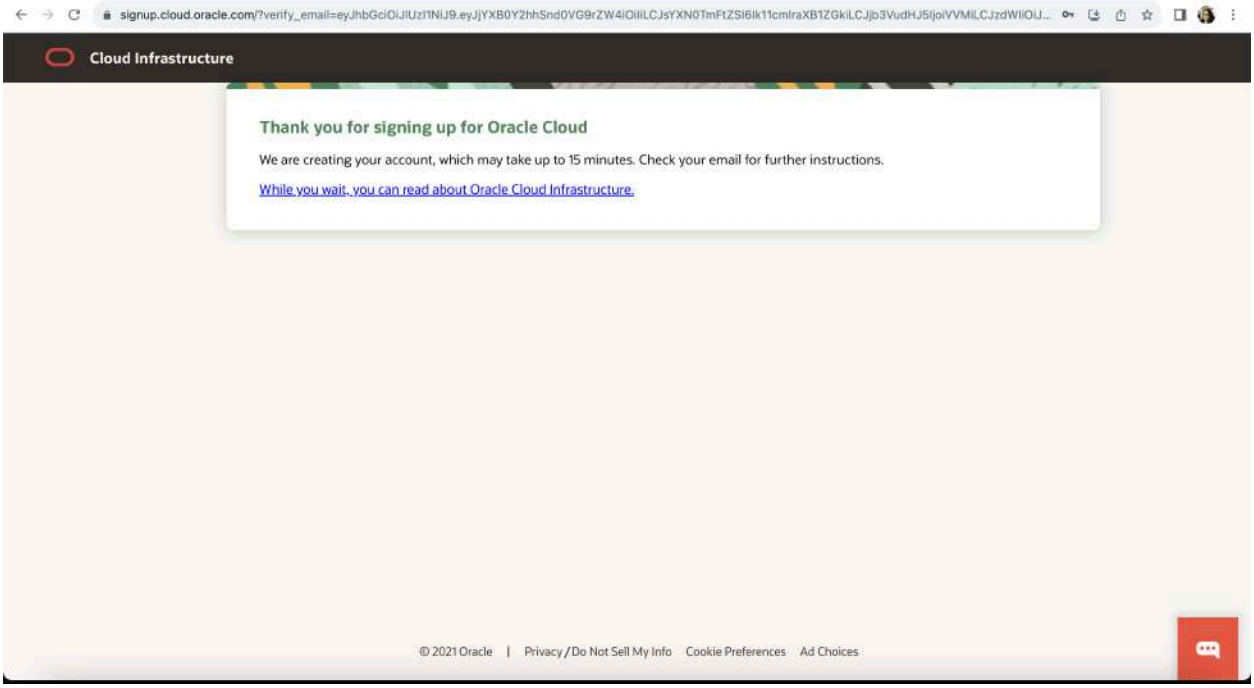

Figure 6: Screenshot of successful sign up to Oracle Cloud.

Once after clicking close, got message saying, “Thank you for signing up for Oracle Cloud”.

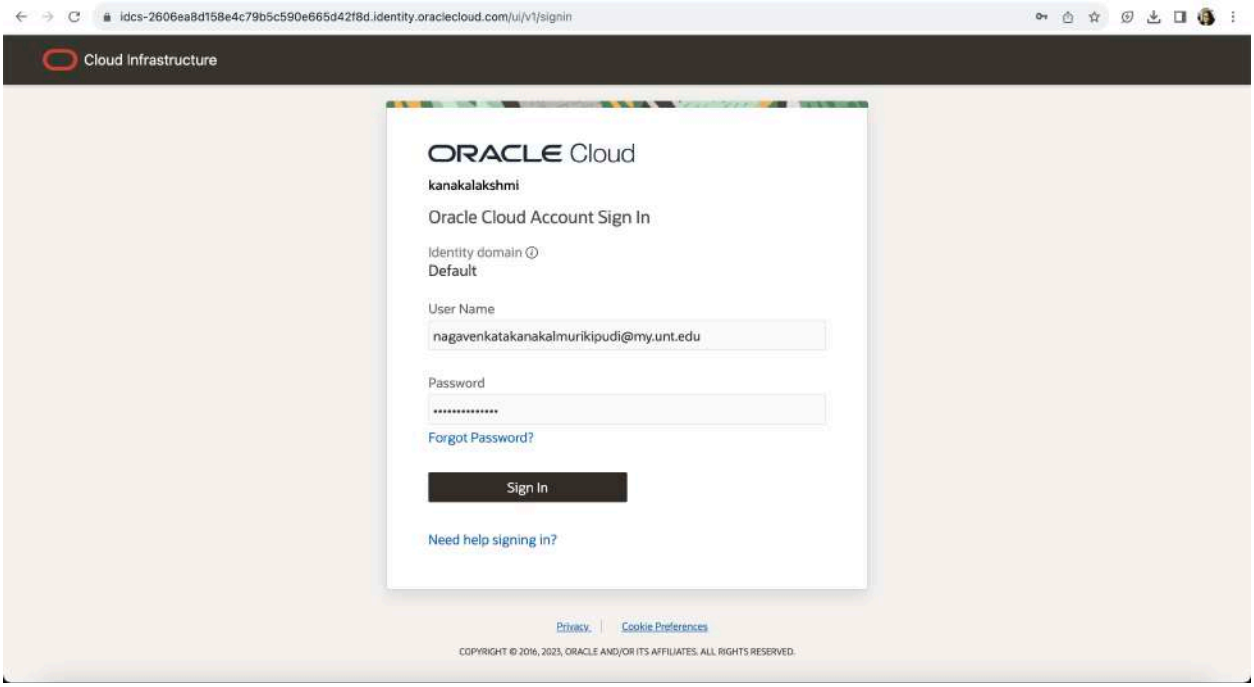

Figure 7: Screenshot of sign in page.

After 15hours of successful sign in, my oracle cloud account got created. I have given Username and Password to sign into oracle cloud.

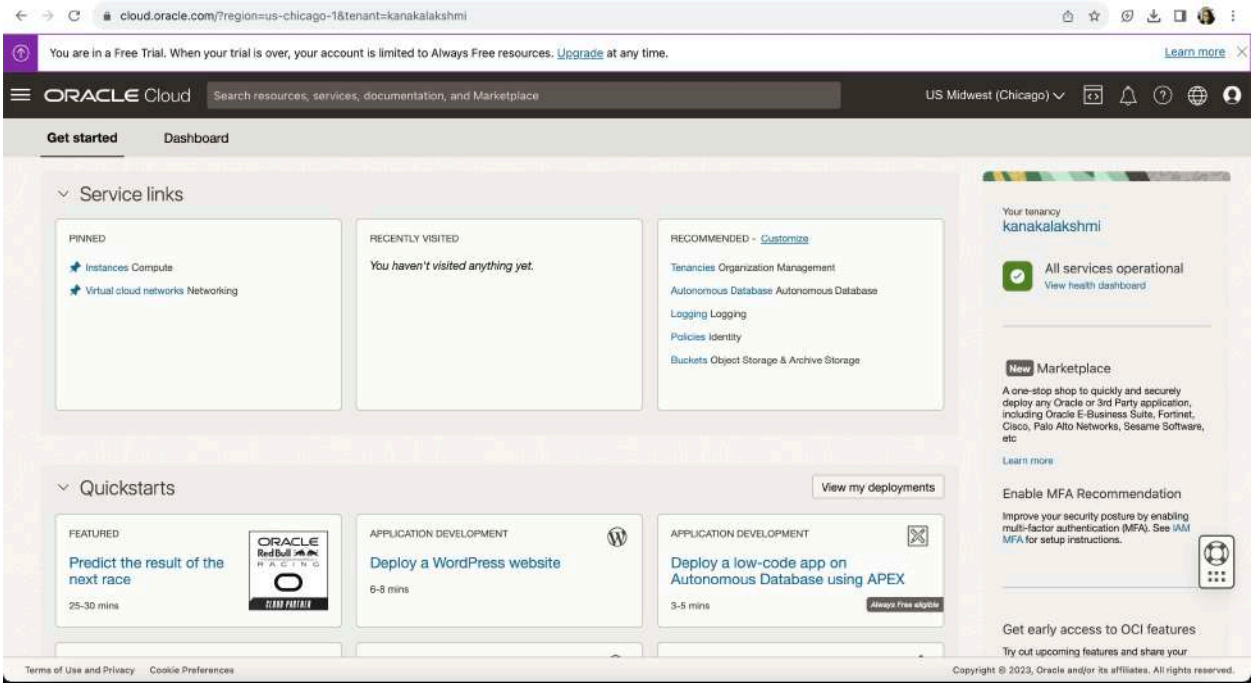

Figure 8: Screenshot of Oracle Cloud Dashboard.

Once after successful sign in, I have seen the respective dashboard.

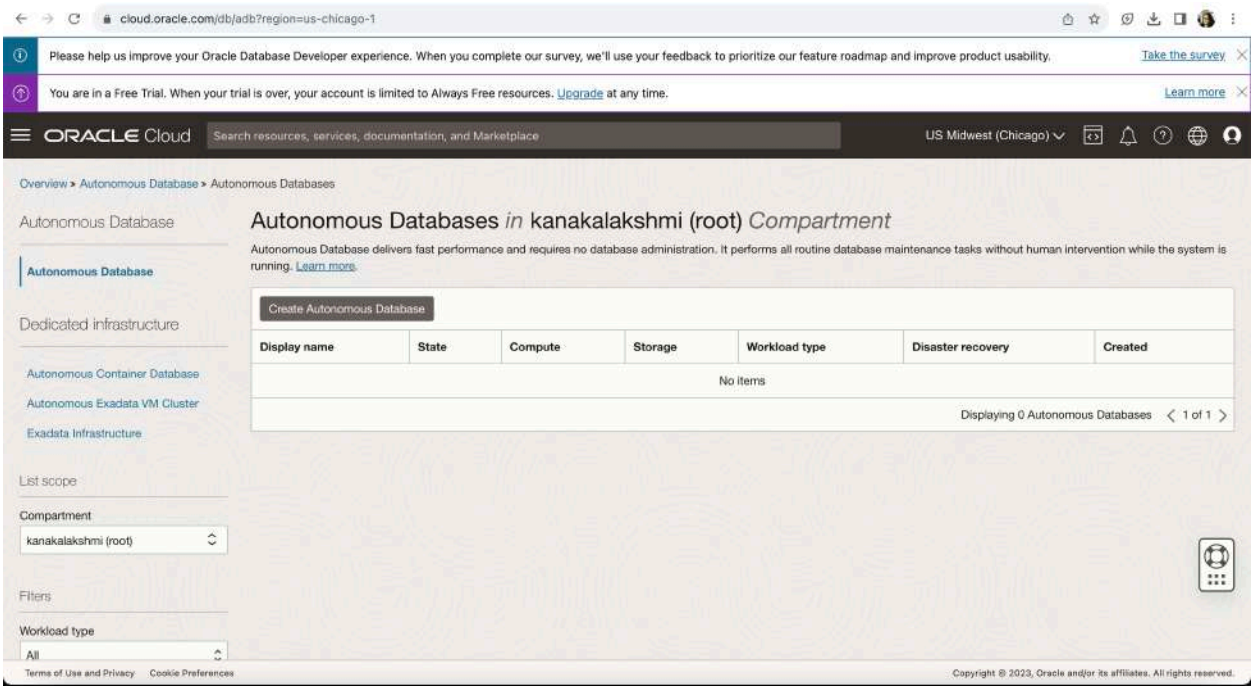

Figure 9: Screenshot of Autonomous Databases page.

We can create autonomous database from this page.

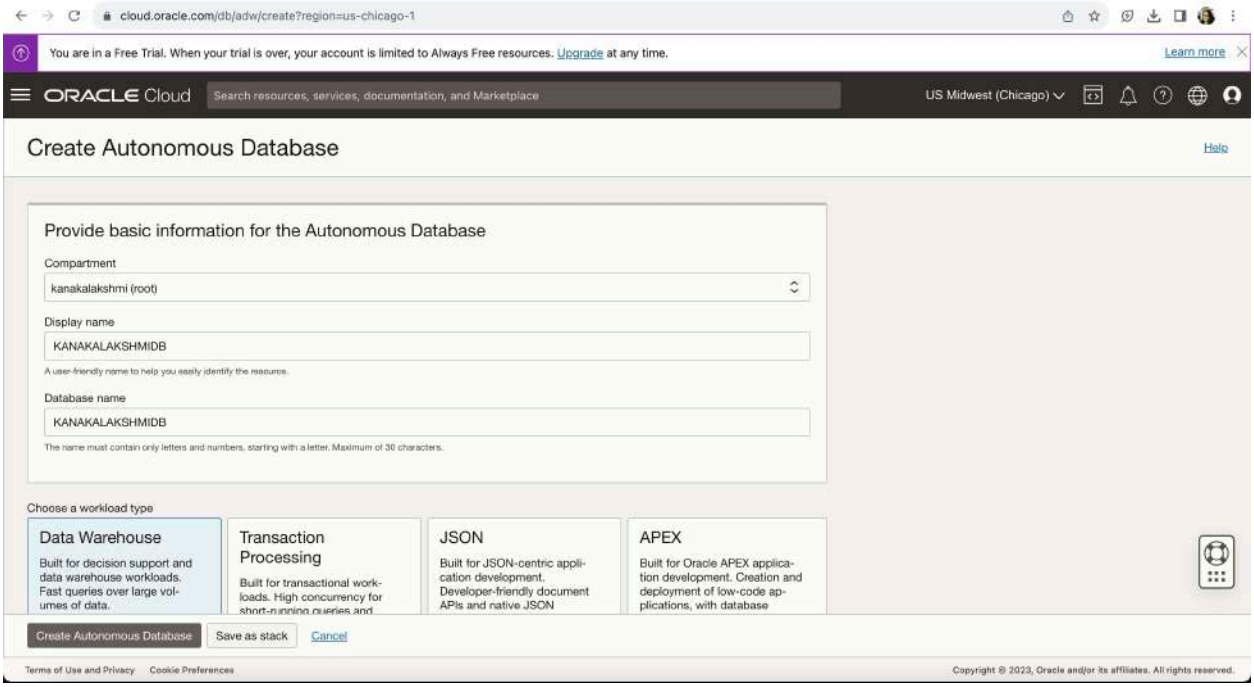

Figure 10: Screenshot of Create Autonomous Database.

In Figure 10, I have given details that is compartment, Display name, Database name to create an autonomous database in oracle cloud.

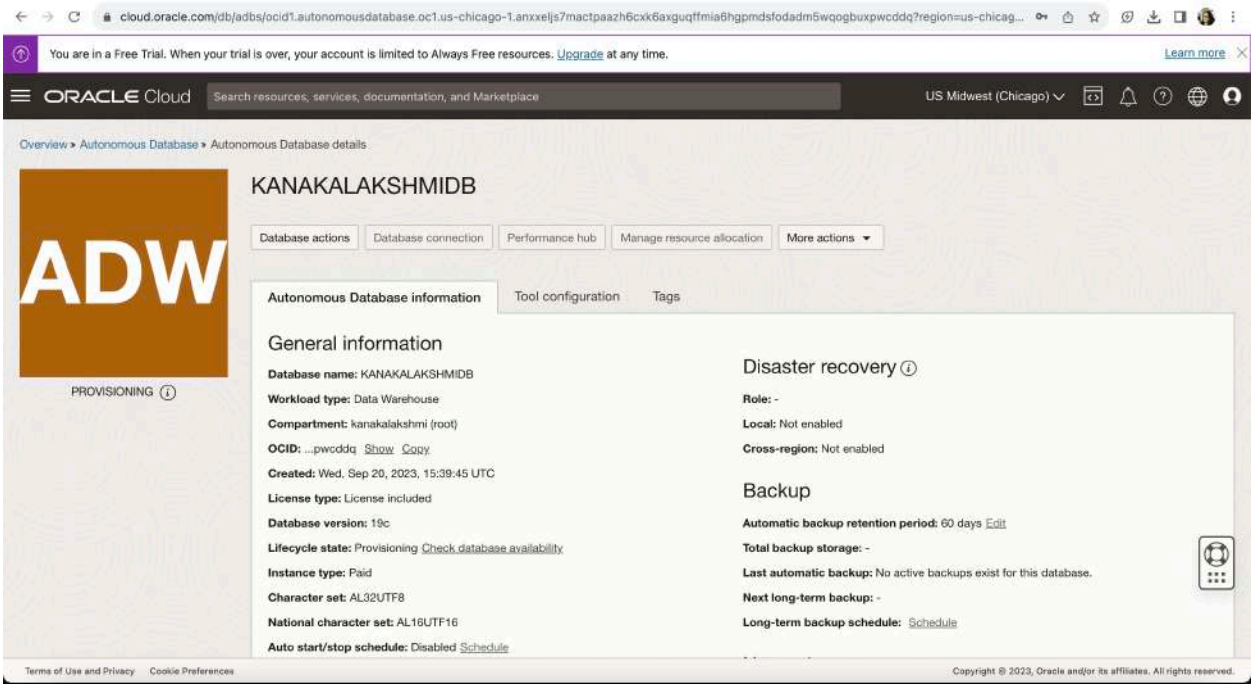

Figure 11: Screenshot of Provisioning.

Once after providing the details, it initiates the Database creation. Figure 11 says that Database creation in “Provisioning”.

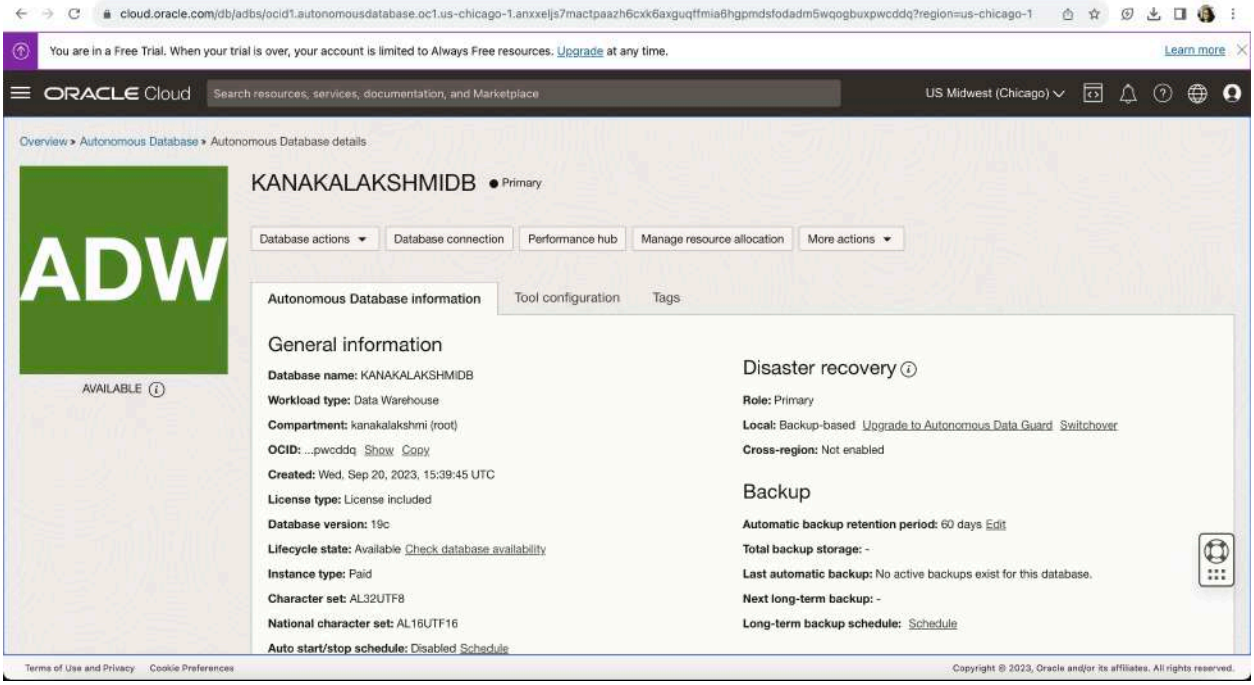

Figure 12: Screenshot of Database Available.

Autonomous Database is created and is in Available state.

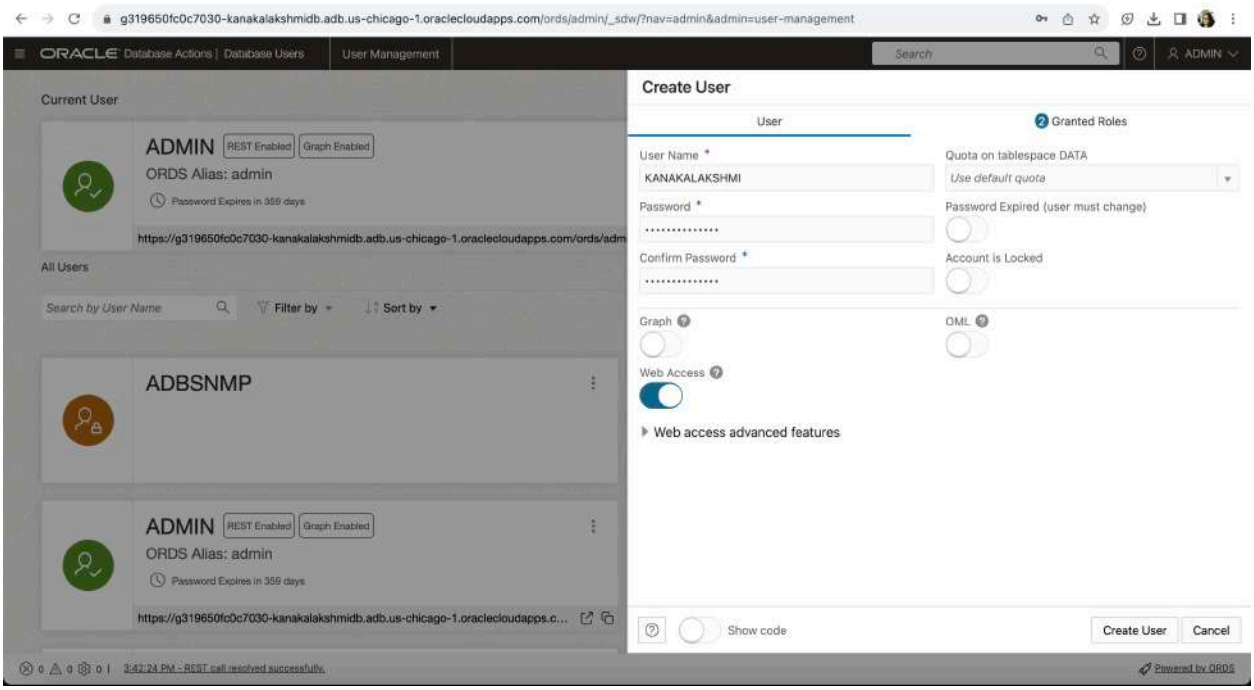

Figure 13: Screenshot of Create user in Autonomous Database

Creating user to access the DB, where we need to provide the username, password, web access is required or not.

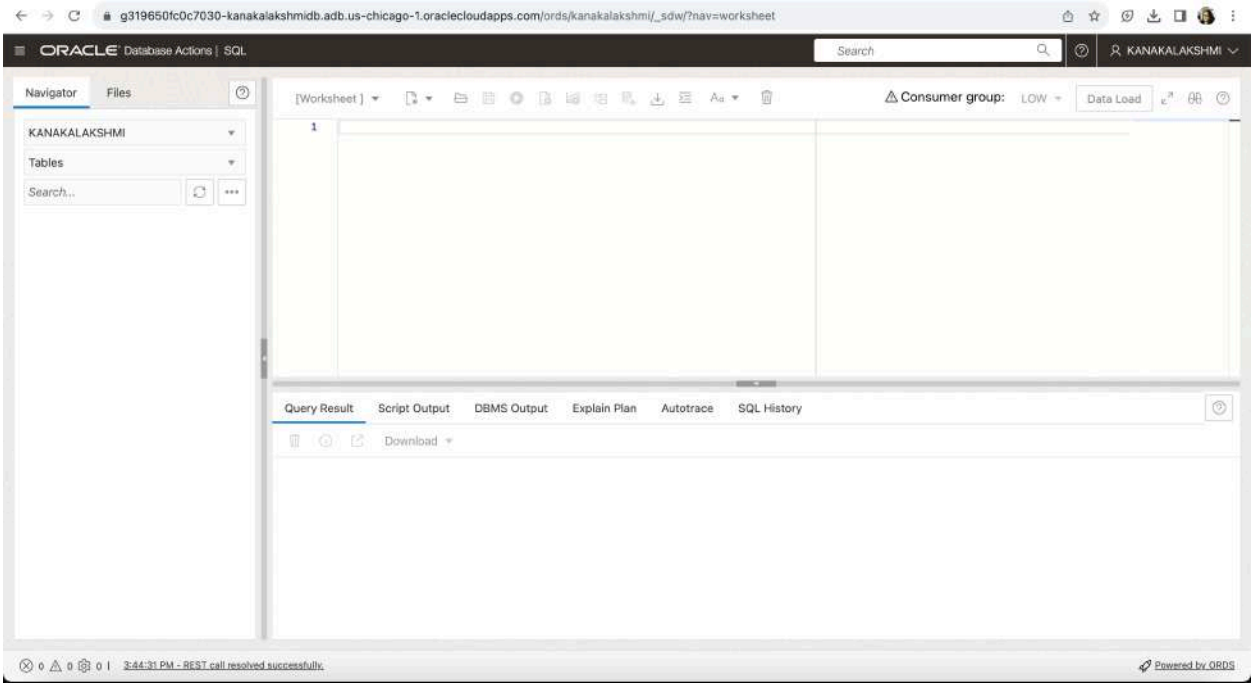

Figure 14: Screenshot of User SQL Worksheet.

Once after creating the Database user, here is the sql worksheet of the user. Where user can start writing the SQL Queries.

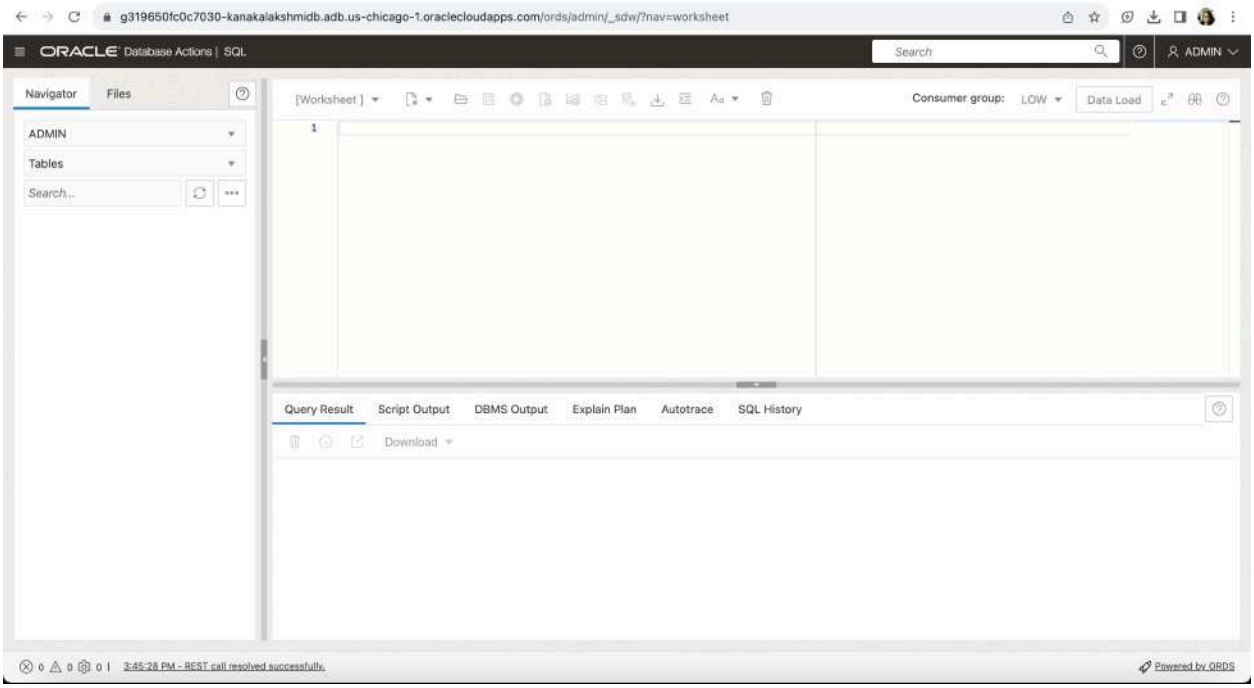

Figure 15: Screenshot of Admin SQL Worksheet.

Whenever we create an Autonomous database, by default there will be an admin user where he can grant permissions to all other users to access the database.

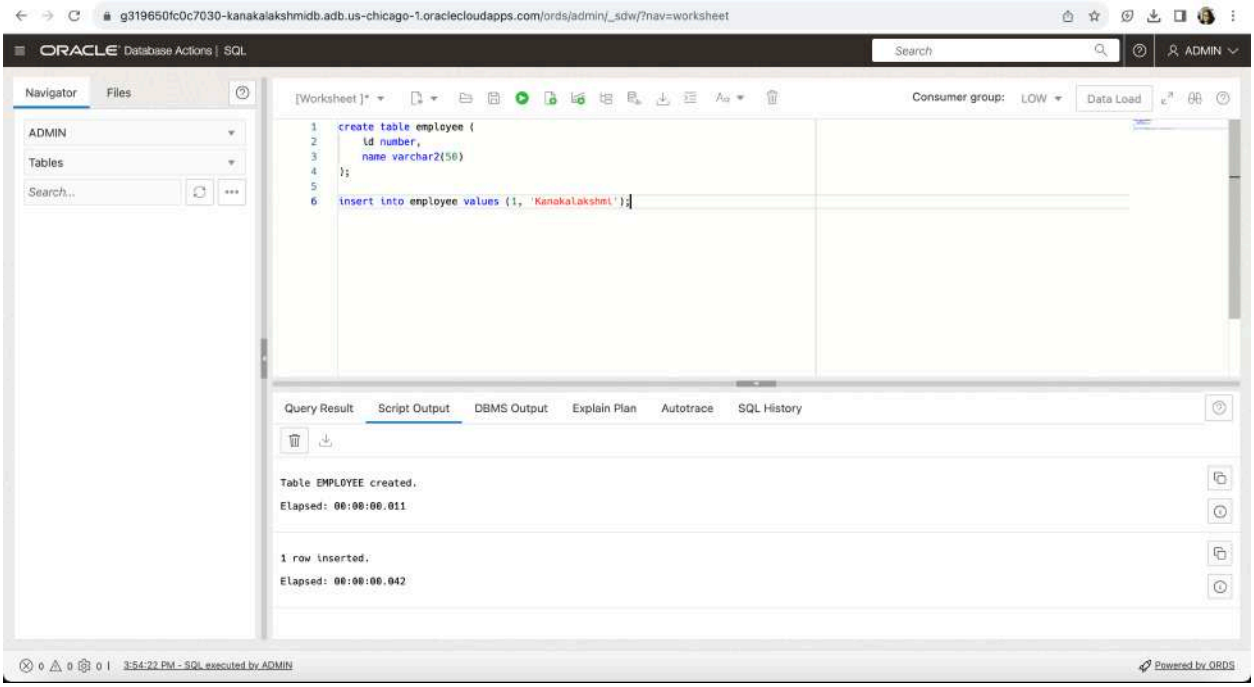

Figure 16: Created and Inserted data in Table.

Created a table employee and inserted a row in employee table through ADMIN user.

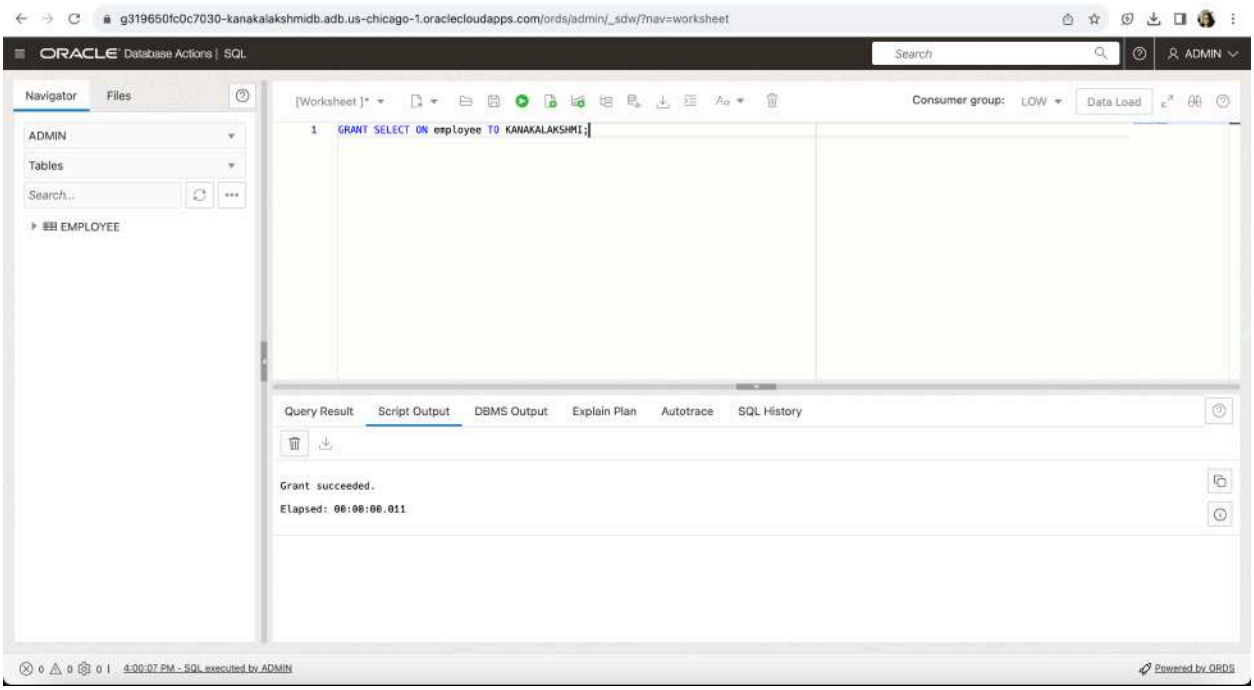

Figure 17: Grant access to User

Granted access to KANAKALAKSHMI USER on employee table.

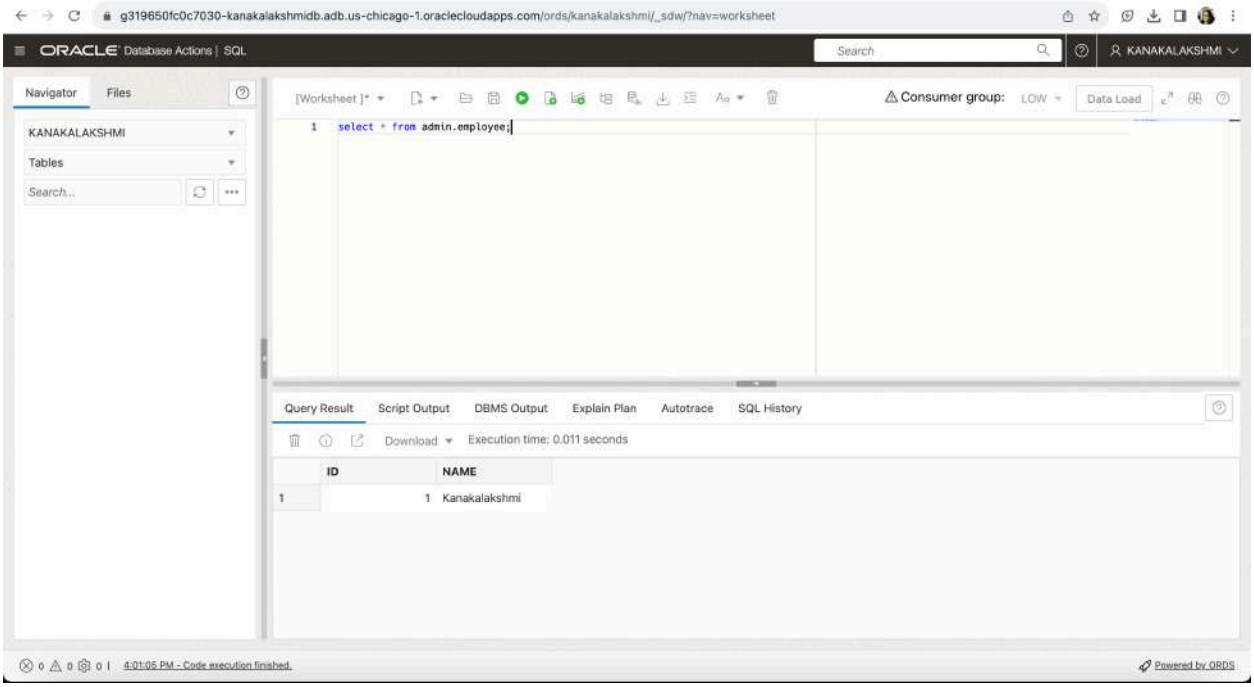

Figure 18: Displayed employee table from “KANAKALAKSHMI” user.

Selected the rows from the employee table created by ADMIN user and accessed by KANAKALAKSHMI user. Which demonstrates the distributed DB environment.

Oracle Local Installation:

I have installed the Oracle Database in Mac OS which runs on intel processor. The following has been used to install the database and access it:

- VirtualBox
- Vagrant
- SQL Developer

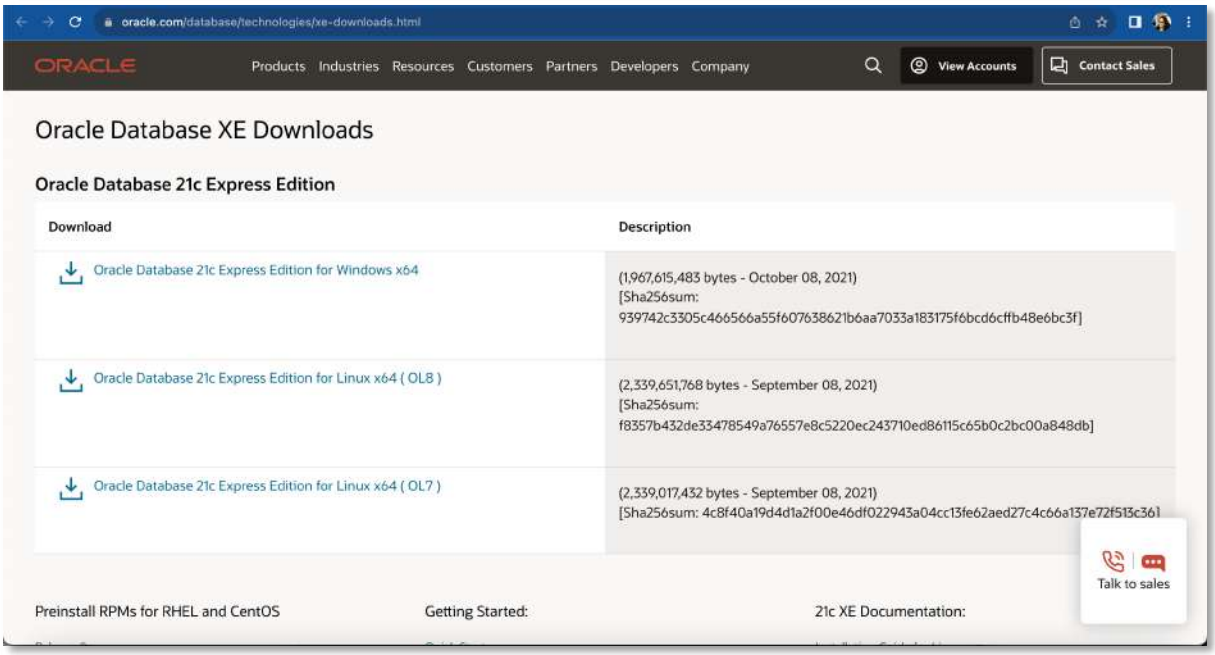

Figure 1: Oracle Downloads page

Figure 1 shows that there is no download module of MAC OSX. So, we using VirtualBox and Vagrant to install it.

VirtualBox Installation:

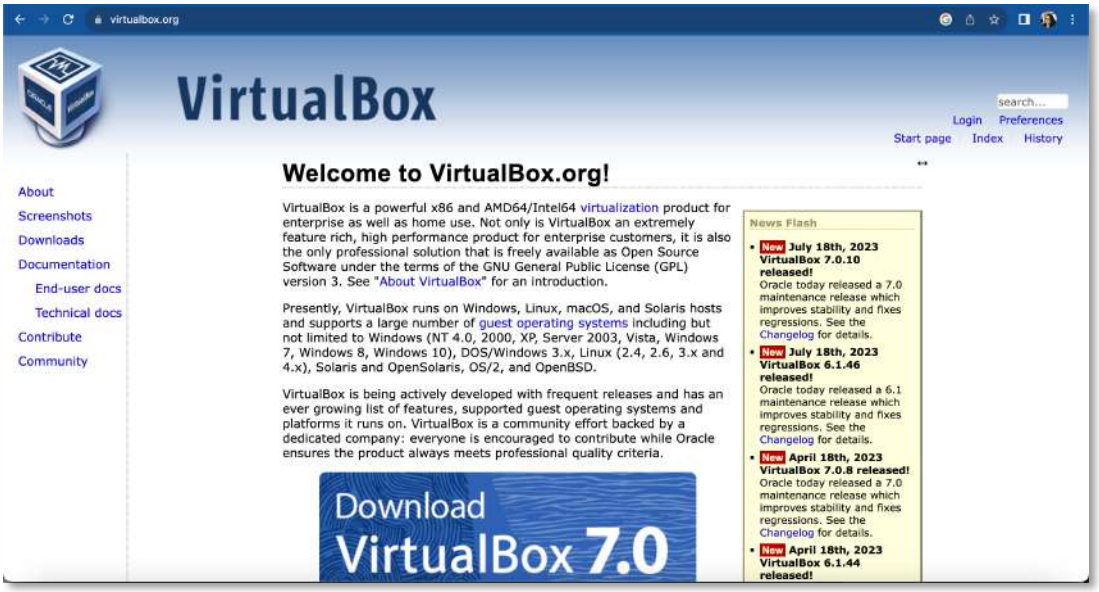

Figure 2: VirtualBox Home page

Figure 2 shows the home page of virtual box, from where we can navigate to downloads page and download it. Now, after moving to downloads page, we can download the latest virtual box.

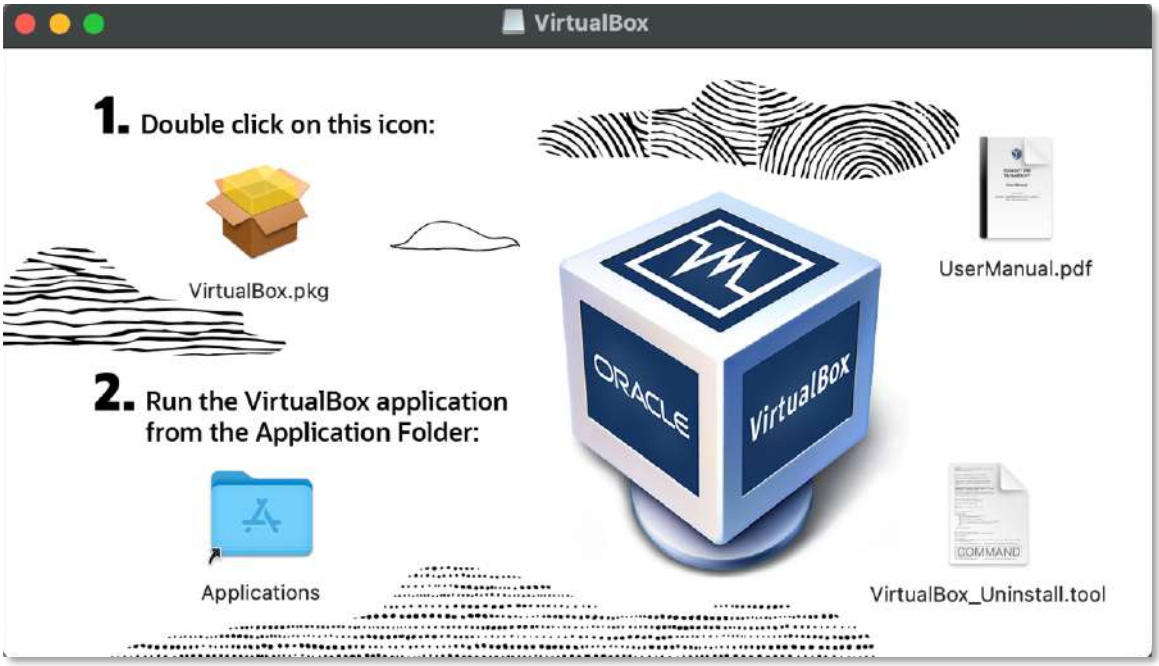

Figure 3: VirtualBox Installation 1

Figure 4: VirtualBox Installation 2

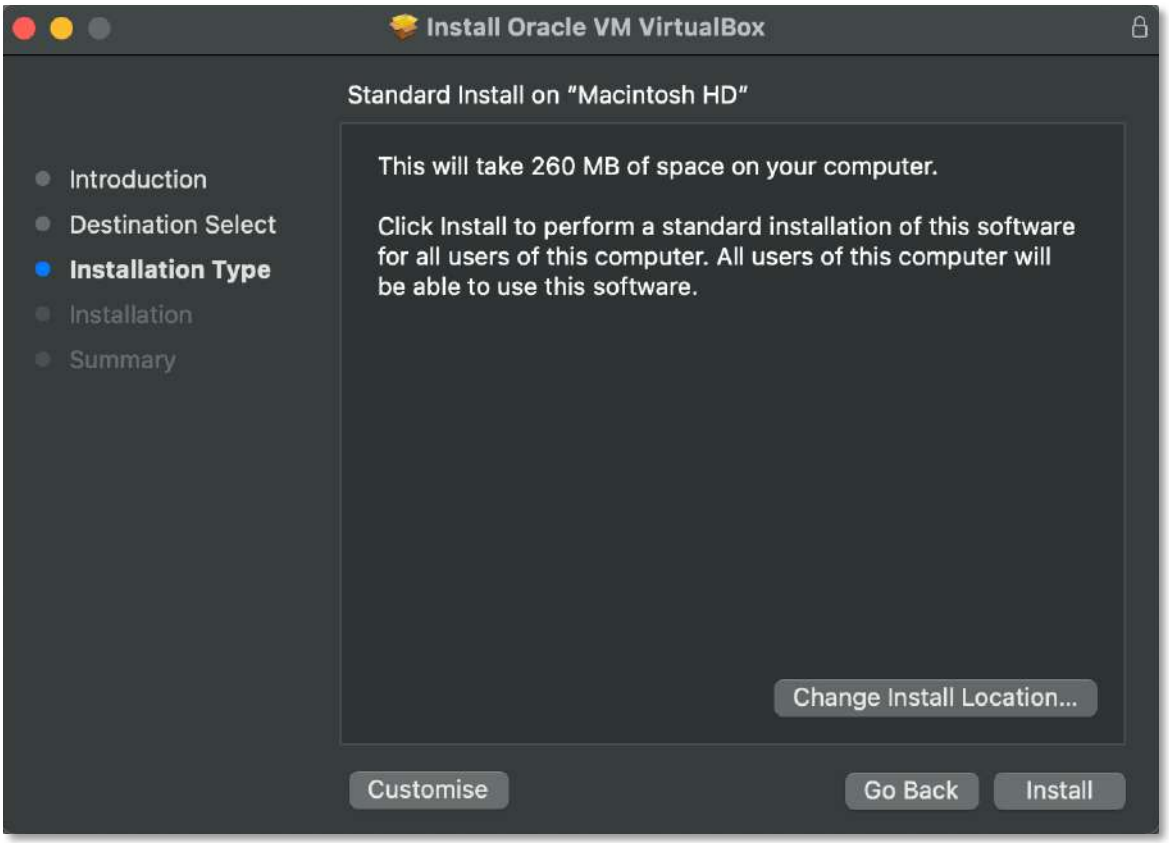

Figure 5: VirtualBox Installation 3

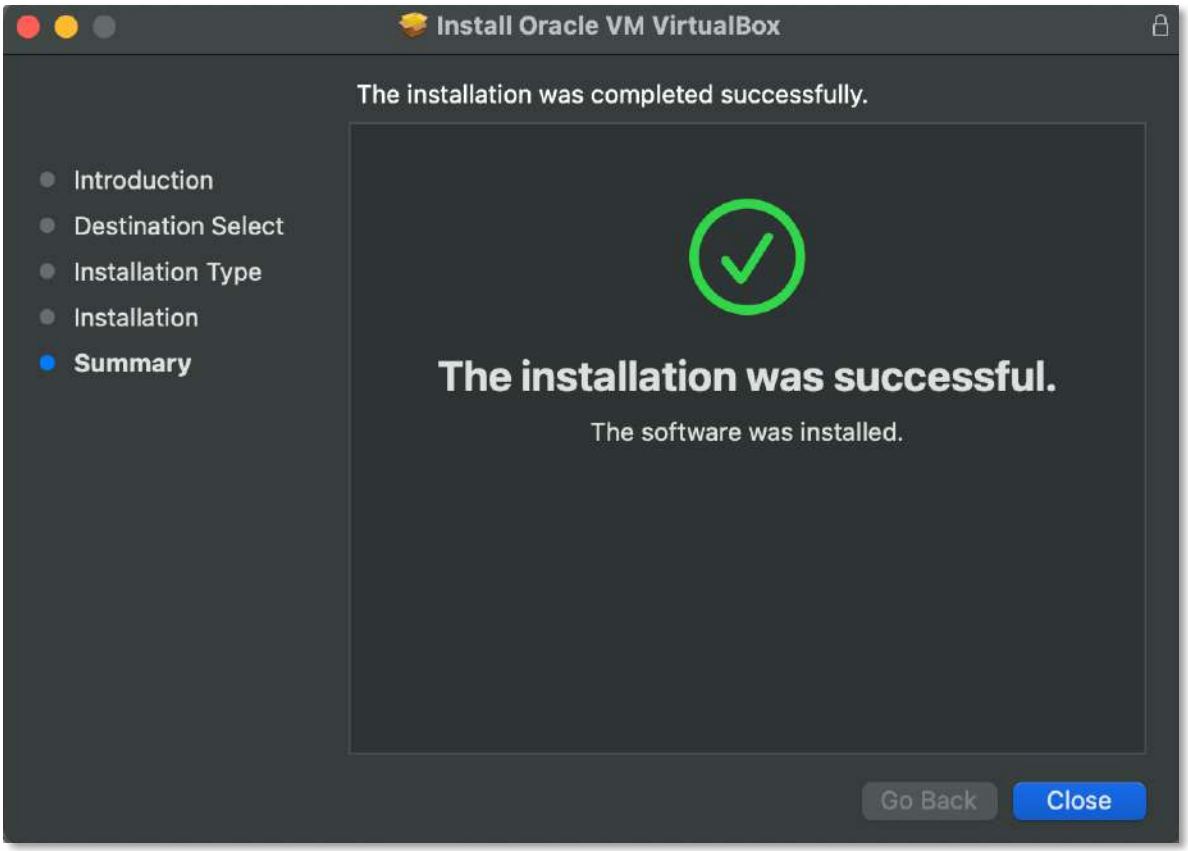

Figure 6: VirtualBox Installation 4

Figure 3, 4, 5, 6 shows the installation process of VirtualBox.

Vagrant Installation:

Now, we need to install “Hashicorp-vagrant”, which is an open-source tool for creating and managing virtual development environments. In this case, we are using to install Oracle.

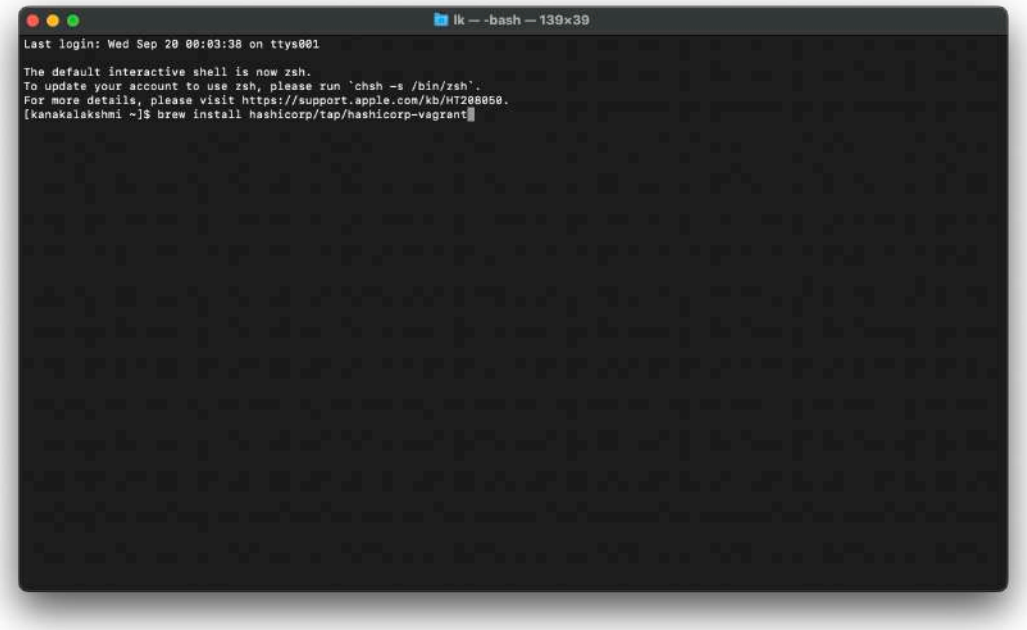

Figure 7: Vagrant installation

Vagrant is installed into Mac OS using “brew” package installer. Figure 7 shows the command to install the “hashicorp-vagrant” using brew installer.

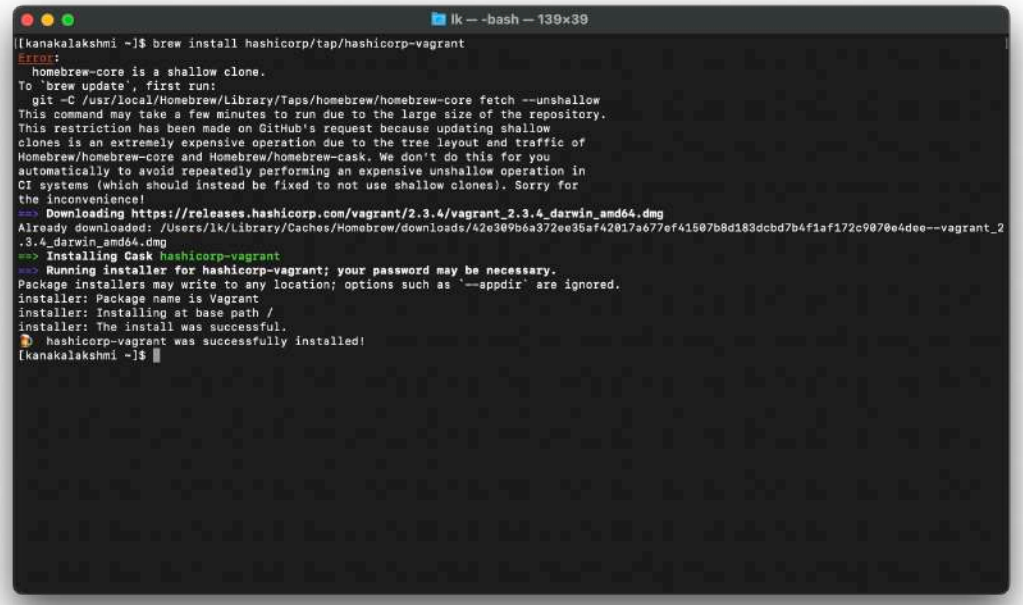

Figure 8: Vagrant installation complete

Figure 8 shows that the vagrant installation is complete.
Now, we need to clone vagrant oracle database images with which we can install oracle database in Mac OS particularly in VirtualBox.

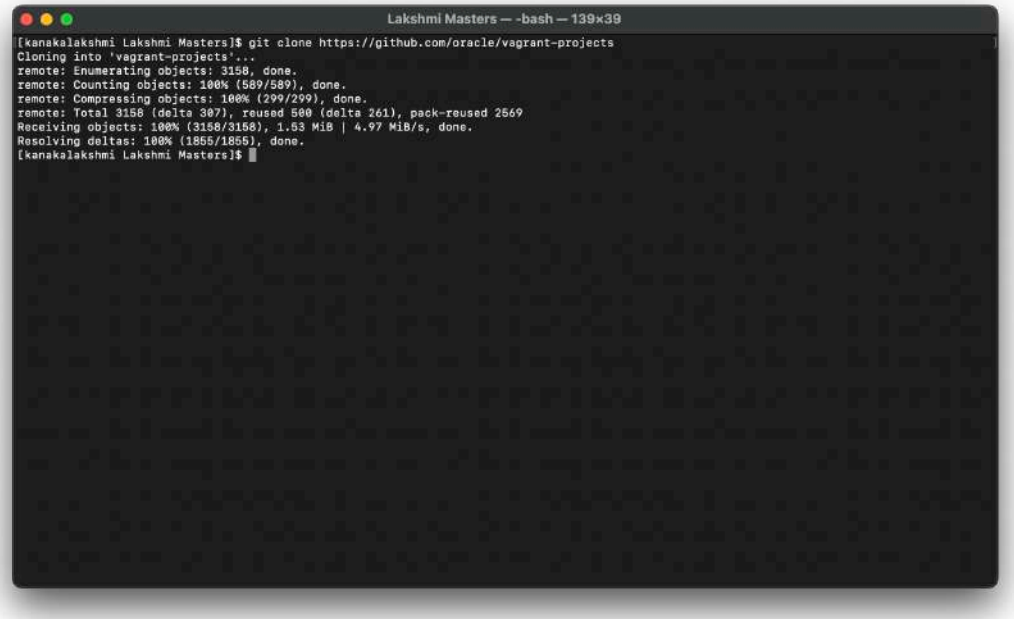

Figure 9: Vagrant oracle images cloned.

Oracle Installation:

Now, we will start the installation process by using vagrant by navigating to the directory where Oracle 21.3.0-XE database image is located.

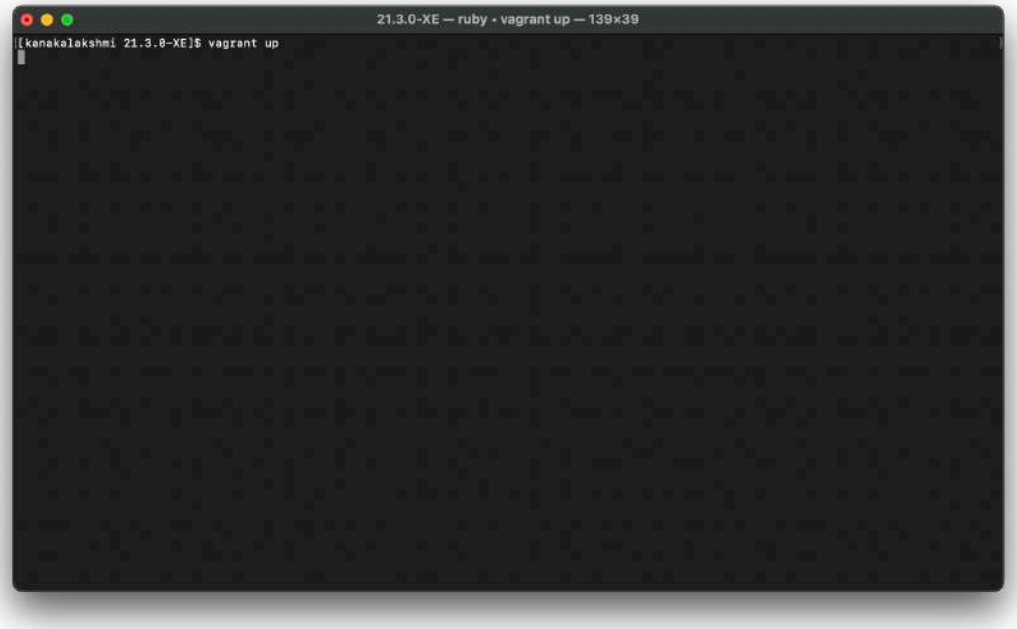

Figure 10: Vagrant up – Oracle installation.

Figure 10 shows the command of vagrant to start the installation of oracle i.e., “vagrant up”

```
21.3.0-XE — ruby ◂ vagrant up — 139×39
oracle21c-xe-vagrant:   lm_sensors-libs-3.4.0-23.20180522git70f7e08.el8.x86_64
oracle21c-xe-vagrant:   net-tools-2.0-0.52.20160912git.el8.x86_64
oracle21c-xe-vagrant:   nfs-utils-1:2.3.3-59.el8.x86_64
oracle21c-xe-vagrant:   oracle-database-preinstall-21c-1.0-1.el8.x86_64
oracle21c-xe-vagrant:   protobuf-c-1.3.0-6.el8.x86_64
oracle21c-xe-vagrant:   python3-bind-32:9.11.36-8.el8_8.1.noarch
oracle21c-xe-vagrant:   python3-ply-3.9-9.el8.noarch
oracle21c-xe-vagrant:   python3-pyyaml-3.12-12.el8.x86_64
oracle21c-xe-vagrant:   quota-1:4.04-14.el8.x86_64
oracle21c-xe-vagrant:   quota-nls-1:4.04-14.el8.noarch
oracle21c-xe-vagrant:   rpcbind-1.2.5-10.el8.x86_64
oracle21c-xe-vagrant:   smartmontools-1:7.1-1.el8.x86_64
oracle21c-xe-vagrant:   sysstat-11.7.3-9.0.1.el8.x86_64
oracle21c-xe-vagrant:   xorg-x11-utils-7.5-28.el8.x86_64
oracle21c-xe-vagrant:   xorg-x11-xauth-1:1.0.9-12.el8.x86_64
oracle21c-xe-vagrant:
oracle21c-xe-vagrant: Complete!
oracle21c-xe-vagrant: INSTALLER: Oracle preinstall and openssl complete
oracle21c-xe-vagrant: INSTALLER: Environment variables set
oracle21c-xe-vagrant: INSTALLER: Downloading Oracle Database software
oracle21c-xe-vagrant: Last metadata expiration check: 0:14:14 ago on Wed 20 Sep 2023 12:23:12 AM -05.
oracle21c-xe-vagrant: Dependencies resolved.
oracle21c-xe-vagrant: ================================================================================
oracle21c-xe-vagrant:  Package                    Architecture Version     Repository          Size
oracle21c-xe-vagrant: ================================================================================
oracle21c-xe-vagrant: Installing:
oracle21c-xe-vagrant:  oracle-database-xe-21c       x86_64       1.0-1       @commandline       2.2 G
oracle21c-xe-vagrant:
oracle21c-xe-vagrant: Transaction Summary
oracle21c-xe-vagrant: ================================================================================
oracle21c-xe-vagrant: Install  1 Package
oracle21c-xe-vagrant:
oracle21c-xe-vagrant: Total size: 2.2 G
oracle21c-xe-vagrant: Installed size: 5.8 G
oracle21c-xe-vagrant: Downloading Packages:
oracle21c-xe-vagrant: Running transaction check
oracle21c-xe-vagrant: Transaction check succeeded.
oracle21c-xe-vagrant: Running transaction test
```

Figure 11: Oracle installation 1.

```
21.3.0-XE — ruby ◂ vagrant up — 139×39
oracle21c-xe-vagrant: Install  1 Package
oracle21c-xe-vagrant:
oracle21c-xe-vagrant: Total size: 2.2 G
oracle21c-xe-vagrant: Installed size: 5.8 G
oracle21c-xe-vagrant: Downloading Packages:
oracle21c-xe-vagrant: Running transaction check
oracle21c-xe-vagrant: Transaction check succeeded.
oracle21c-xe-vagrant: Running transaction test
oracle21c-xe-vagrant: Transaction test succeeded.
oracle21c-xe-vagrant: Running transaction
oracle21c-xe-vagrant:   Preparing        :                                                    1/1
oracle21c-xe-vagrant:   Running scriptlet: oracle-database-xe-21c-1.0-1.x86_64                1/1
oracle21c-xe-vagrant:   Installing       : oracle-database-xe-21c-1.0-1.x86_64                1/1
oracle21c-xe-vagrant:   Running scriptlet: oracle-database-xe-21c-1.0-1.x86_64                1/1
oracle21c-xe-vagrant: [INFO] Executing post installation scripts...
oracle21c-xe-vagrant: [INFO] Oracle home installed successfully and ready to be configured.
oracle21c-xe-vagrant: To configure Oracle Database XE, optionally modify the parameters in '/etc/sysconfig/oracle-xe-21c.conf' and then
execute '/etc/init.d/oracle-xe-21c configure' as root.
oracle21c-xe-vagrant:
oracle21c-xe-vagrant:   Verifying        : oracle-database-xe-21c-1.0-1.x86_64                1/1
oracle21c-xe-vagrant:
oracle21c-xe-vagrant: Installed:
oracle21c-xe-vagrant:   oracle-database-xe-21c-1.0-1.x86_64
oracle21c-xe-vagrant:
oracle21c-xe-vagrant: Complete!
oracle21c-xe-vagrant: INSTALLER: Oracle software installed
oracle21c-xe-vagrant: Configuring Oracle Listener.
oracle21c-xe-vagrant: Listener configuration succeeded.
oracle21c-xe-vagrant: Configuring Oracle Database XE.
oracle21c-xe-vagrant: Enter SYS user password:
oracle21c-xe-vagrant: *************
oracle21c-xe-vagrant: Enter SYSTEM user password:
oracle21c-xe-vagrant: ***************
oracle21c-xe-vagrant: Enter PDBADMIN User Password:
oracle21c-xe-vagrant: ***************
oracle21c-xe-vagrant: Prepare for db operation
oracle21c-xe-vagrant: 7% complete
oracle21c-xe-vagrant: Copying database files
```

Figure 12: Oracle installation 2.

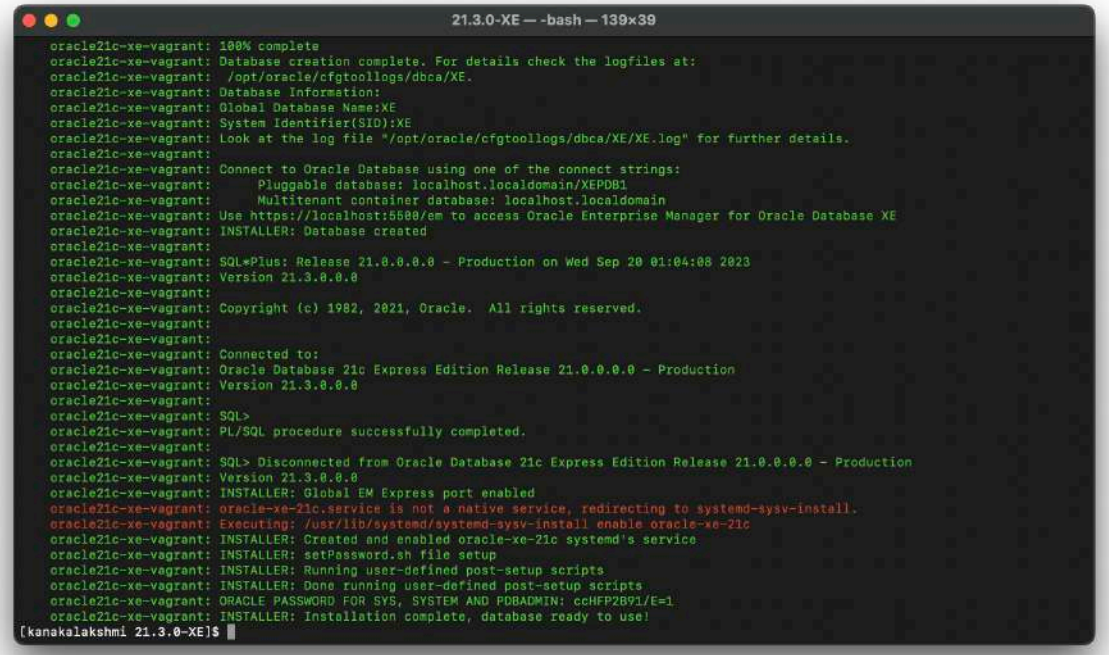

Figure 13: Oracle installation 3.

Figure 11, 12, 13 shows the oracle installation process step by step. This step takes some time as we are installing the oracle for the first time. Once it is setup, it will second to make the oracle up using the same command. In the end, we can see that “Installation done, DB is ready to use”.

SQL Developer Installation:

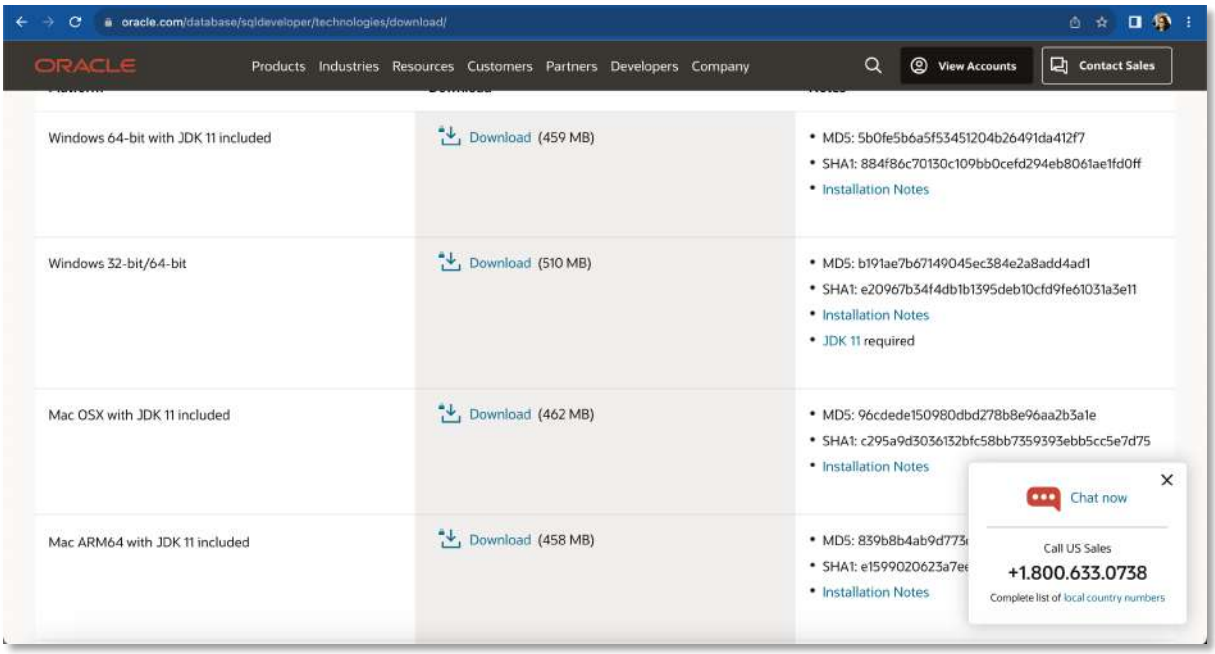

Figure 14: SQL Developer downloads page

Figure 14 shows the downloads page of SQL Developer, which can be used to connect to any SQL databases.

Figure 15: SQL Developer.

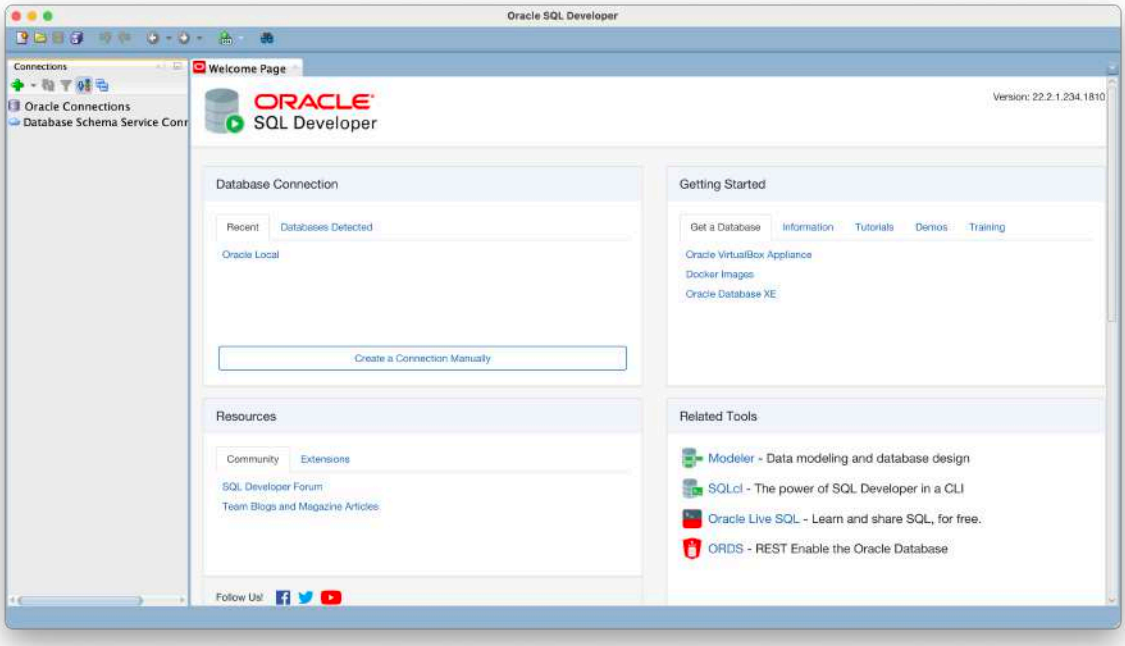

Figure 16: SQL Developer home page.

Figure 16 shows the homepage of the SQL Developer, where we can start connecting to Oracle DB and perform actions.

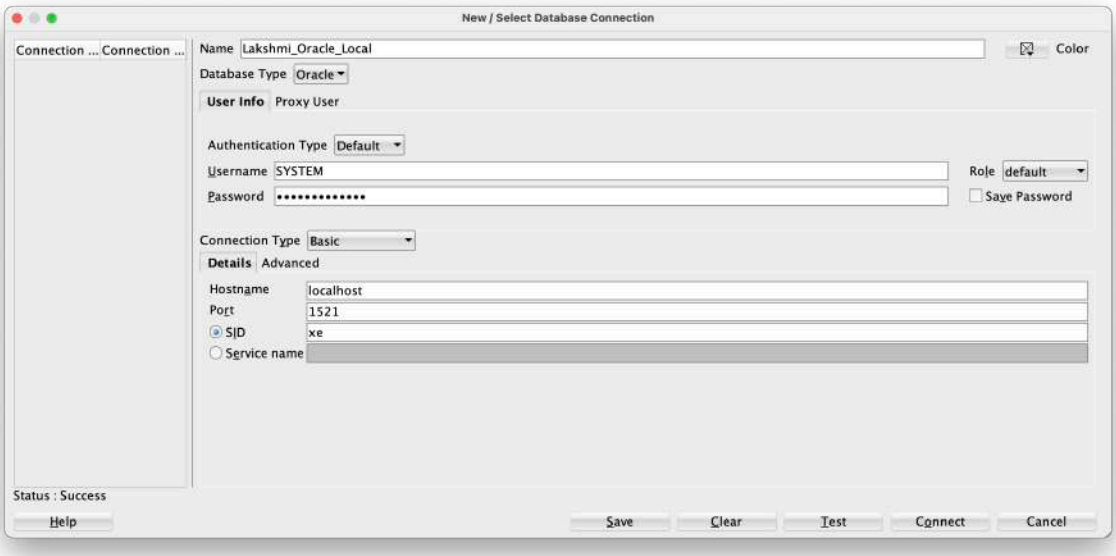

Figure 17: SQL Developer connection page.

Figure 17 shows the connection page of SQL developer, where we can provide the username, password, host, port etc.. to connect to a SQL DB. By clicking on the “Test”, we have tested the connection which is a success.

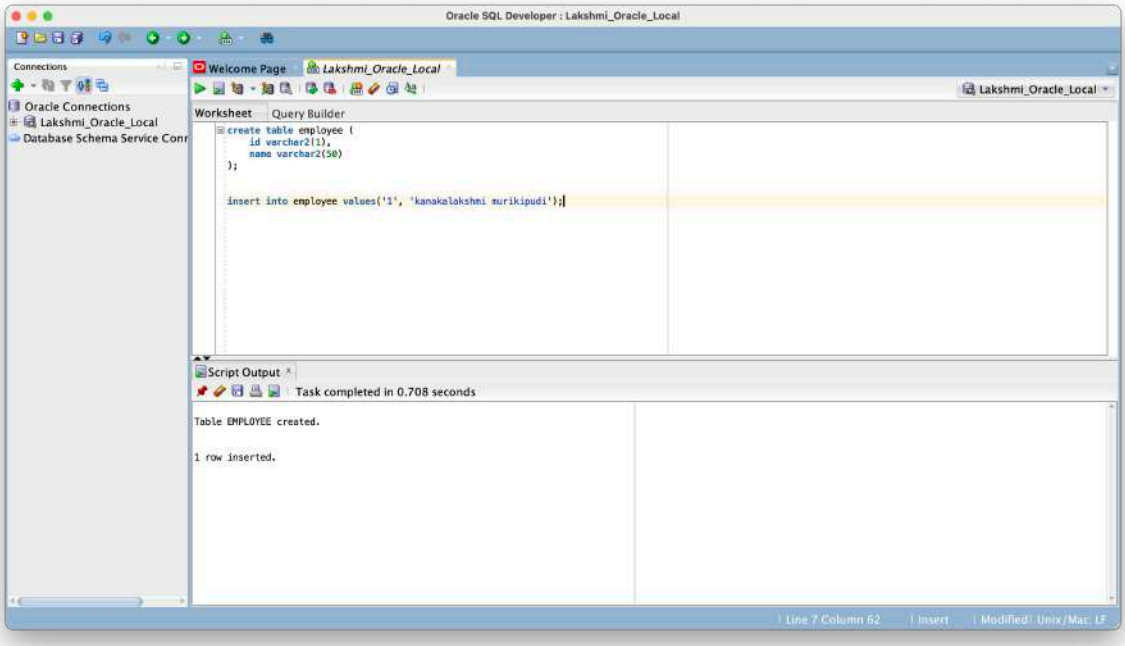

Figure 18: SQL Developer worksheet page.

Figure 18 shows the creation of a oracle table “employee” and insertion of a record into it. This shows a successful connection to oracle database.

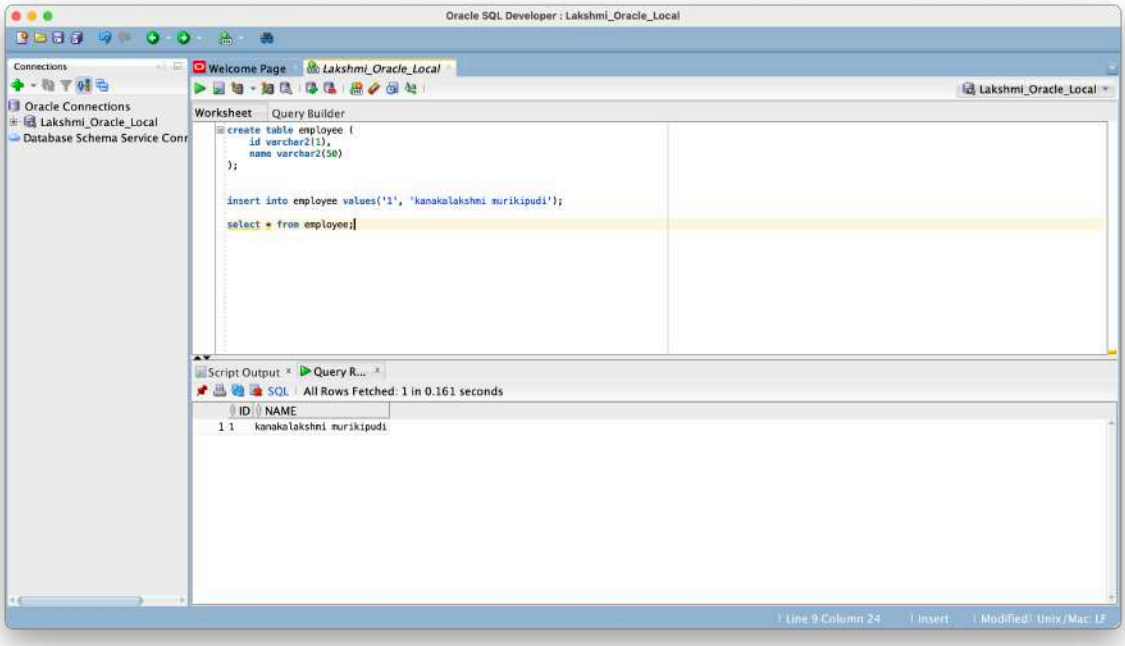

Figure 19: SQL Developer worksheet page.

Figure 19 shows the selection of a oracle table “employee”.

Analysis

Table	Attributes	Domain Contraints	Contraints	Constraint Name	Relations
automotive_retailer	id	RAW(16)	PRIMARY KEY	NA	automotive_retailer has one to many relationship with employee, automotive_retailer has one to many relationship with inventory_product, automotive_retailer has one to one relationship with address, automotive_retailer has one to many relationship with job
	phone	VARCHAR2(20)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (phone, '^\ (\d{3}\) \d{3}-\d {4}$'))	invalid_automotive_retailer_phone	
	email	VARCHAR2(255)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (email, '^[A-Za-z0-9. _%+-]+@[A-Za-z0-9.-] +\.[A-Za-z]{2,}$'))	invalid_automotive_retailer_email	
	website	VARCHAR2(255)	NOT NULL, UNIQUE	NA	
	business_hours	VARCHAR2(20)	NOT NULL	NA	
	manager_id	RAW(16)	NOT NULL, FOREIGN KEY (manager_id) REFERENCES employee(id)	NA	
	address_id	RAW(16)	NOT NULL, FOREIGN KEY (address_id) REFERENCES address(id)	NA	
inventory_product	id	RAW(16)	PRIMARY KEY	NA	inventory_product has many to many relationship with supplier, inventory_product has many to one realationship with automotive_retailer, inventory_product has many to many relationship with bill, inventory_product has one to many relationship with part_service, inventory_product has many to one relationship with automobile, inventory_product has many to one relationship with address
	name	VARCHAR2(100)	NOT NULL	NA	
	quantity	INTEGER	NOT NULL	NA	
	price	INTEGER	NOT NULL	NA	
	automotive_retailer_id	RAW(16)	NOT NULL, FOREIGN KEY (automotive_retailer_id) REFERENCES automotive_retailer(id)	NA	
	automobile_id	RAW(16)	"NOT NULL, FOREIGN KEY (automobile_id) REFERENCES automobile(id)"	NA	
	address_id	RAW(16)	NOT NULL, FOREIGN KEY (address_id) REFERENCES address(id)	NA	
automobile	id	RAW(16)	PRIMARY KEY	NA	automobile has one to many relationship with inventory_product, automobile has one to many relationship with job
	manufacturer	VARCHAR2(100)	NOT NULL	NA	
	name	VARCHAR2(100)	NOT NULL	NA	
	variant	VARCHAR2(100)	NOT NULL	NA	
	year	VARCHAR2(10)	NOT NULL	NA	
	color	VARCHAR2(100)	NOT NULL	NA	
supplier	id	RAW(16)	PRIMARY KEY	NA	supplier has many to many relationship with inventory_product, supplier has one to one relationship with address
	name	VARCHAR2(100)	NOT NULL	NA	
	phone	VARCHAR2(20)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (phone, '^\ (\d{3}\) \d{3}-\d {4}$'))	invalid_supplier_phone	

Analysis

Table	Attributes	Domain Contraints	Contraints	Constraint Name	Relations
	address_id	RAW(16)	NOT NULL, FOREIGN KEY (address_id) REFERENCES address(id)	NA	
Employee	id	RAW(16)	PRIMARY KEY	NA	employee has one to many relationship with employee_payroll, employee has many to one relationship with automotive_retailer, employee has one to many relationship with bill, employee has many to one relationship with address, employee has many to many relationship with job
	first_name	VARCHAR2(100)	NOT NULL	NA	
	last_name	VARCHAR2(100)	NOT NULL	NA	
	dob	TIMESTAMP	NOT NULL	NA	
	phone	VARCHAR2(20)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (phone, '^\ (\d{3}\) \d{3}-\d {4}$'))	invalid_employee_phone	
	email	VARCHAR2(255)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (email, '^[A-Za-z0-9. _%+-]+@[A-Za-z0-9.-] +\.[A-Za-z]{2,}$'))	invalid_employee_email	
	annual_salary	INTEGER	NOT NULL	NA	
	ssn	CHAR(9)	NOT NULL, UNIQUE	NA	
	automotive_retailer_id	RAW(16)	NOT NULL, FOREIGN KEY (automotive_retailer_id) REFERENCES automotive_retailer(id)	NA	
	address_id	RAW(16)	NOT NULL, FOREIGN KEY (address_id) REFERENCES address(id)	NA	
	hire_date	TIMESTAMP	NOT NULL	NA	
employee_payroll	id	RAW(16)	PRIMARY KEY	NA	employee_payroll has many to one relationship with employee
	hours_worked	NUMBER	NOT NULL	NA	
	start_date	TIMESTAMP	NOT NULL	NA	
	end_date	TIMESTAMP	NOT NULL	NA	
	pay	NUMBER	NOT NULL	NA	
	employee_id	RAW(16)	NOT NULL, FOREIGN KEY (employee_id) REFERENCES employee(id)	NA	
address	id	RAW(16)	PRIMARY KEY	NA	address has one to many relationship with employee, address has one to many relationship with inventory_product, address has one to one relationship with automotive_retailer, address has one to one relationship with supplier, address has one to many relationship with customer
	apartment_no	NUMBER	NOT NULL	NA	
	street	VARCHAR2(100)	NOT NULL	NA	
	city	VARCHAR2(100)	NOT NULL	NA	
	state	VARCHAR2(2)	NOT NULL	NA	
	country	VARCHAR2(50)	NOT NULL	NA	
	zip	CHAR(5)	NOT NULL	NA	
	id	RAW(16)	PRIMARY KEY	NA	

Analysis

Table	Attributes	Domain Contraints	Contraints	Constraint Name	Relations
bill	date	TIMESTAMP	NOT NULL	NA	bill has many to one relationship with employee, bill has many to many relationship with inventory_product, bill has many to one relationship with insurance, bill has many to one relationship with customer, bill has one to one relationship with job, bill has many to one relationship with payment_plan
	mode_of_payment	CHAR(4)	NOT NULL, CHECK (mode_of_payment IN ('CARD', 'CASH'))	invalid_mode_of_payment	
	insurance_id	RAW(16)	NOT NULL, FOREIGN KEY (insurance_id) REFERENCES insurance(id)	NA	
	customer_id	RAW(16)	NOT NULL, FOREIGN KEY (customer_id) REFERENCES customer(id)	NA	
	job_id	RAW(16)	NOT NULL, FOREIGN KEY (job_id) REFERENCES job(id)	NA	
	employee_id	RAW(16)	NOT NULL, FOREIGN KEY (employee_id) REFERENCES employee(id)	NA	
	sale_type	CHAR(7)	NOT NULL, CHECK (mode_of_payment IN ('ONLINE', 'OFFLINE'))	invalid_sale_type	
	payment_plan_id	RAW(16)	NOT NULL, FOREIGN KEY (payment_plan_id) REFERENCES payment_plan(id)	NA	
job	id	RAW(16)	PRIMARY KEY	NA	job has one to one realtionship with bill, job has many to one realtionship with automobile, job has many to one realtionship with automotive_retailer, job has many to many relationship with employee, job has one to many realtionship with part_service, job has many to one realtionship with customer
	date	TIMESTAMP	NOT NULL	NA	
	description	VARCHAR2(255)	NOT NULL	NA	
	customer_id	RAW(16)	NOT NULL, FOREIGN KEY (customer_id) REFERENCES customer(id)	NA	
	automotive_retailer_id	RAW(16)	NOT NULL, FOREIGN KEY (automotive_retailer_id) REFERENCES automotive_retailer(id)	NA	
	vin_number	VARCHAR2(100)	NOT NULL	NA	
	automobile_id	RAW(16)	NOT NULL, FOREIGN KEY (automobile_id) REFERENCES automobile(id)	NA	
part_service	id	RAW(16)	PRIMARY KEY	NA	part_service has many to one relationship with job, part_service has one to one relationship with inventary_product
	name	VARCHAR2(100)	NOT NULL	NA	
	quantity	INTEGER	NOT NULL	NA	
	job_id	RAW(16)	NOT NULL, FOREIGN KEY (job_id) REFERENCES job(id)	NA	
	inventory_product_id	RAW(16)	NOT NULL, FOREIGN KEY (inventory_product_id) REFERENCES inventory_product(id)	NA	

Analysis

Table	Attributes	Domain Contraints	Contraints	Constraint Name	Relations
	type	CHAR(7)	NOT NULL, CHECK (mode_of_payment IN ('SERVICE', 'PART'))	invalid_part_service_type	
payment_plan	id	RAW(16)	PRIMARY KEY	NA	payment_plan has one to many relationship with bill
	name	VARCHAR2(100)	NOT NULL	NA	
	installments	INTEGER	NOT NULL	NA	
	interest	INTEGER	NOT NULL	NA	
customer	id	RAW(16)	PRIMARY KEY	NA	customer has many to many relationship with insurance, customer has one to many relationship with job, customer has many to one relationship with address, customer has one to many relationship with bills
	first_name	VARCHAR2(100)	NOT NULL	NA	
	last_name	VARCHAR2(100)	NOT NULL	NA	
	dob	TIMESTAMP	NOT NULL	NA	
	driverlicense	VARCHAR2(50)	UNIQUE, NOT NULL	NA	
	phone	VARCHAR2(20)	NOT NULL, UNIQUE, CHECK (REGEXP_LIKE (phone, '^\ (\d{3}\) \d{3}-\d {4}$'))	invalid_customer_phone	
	address_id	RAW(16)	NOT NULL, FOREIGN KEY (address_id) REFERENCES address(id)	NA	
insurance	id	RAW(16)	PRIMARY KEY	NA	insurance has many to many relationship with customer, insurance has one to many relationship with bill
	policy_type	VARCHAR2(100)	NOT NULL	NA	
	provider	VARCHAR2(100)	NOT NULL	NA	
	claim_percentage	INTEGER	NOT NULL	NA	
inventory_product_supplier	inventory_product_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (inventory_product_id) REFERENCES inventory_product(id)"	NA	NA
	supplier_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (supplier_id) REFERENCES supplier(id)	NA	
	quantity	INTEGER	NOT NULL	NA	
	bill_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (bill_id) REFERENCES bill(id)"	NA	

Analysis

Table	Attributes	Domain Contraints	Contraints	Constraint Name	Relations
bill_inventory_product	inventory_product_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (inventory_product_id) REFERENCES inventory_product(id)"	NA	NA
	quantity	INTEGER	NOT NULL	NA	
job_employee	job_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (job_id) REFERENCES job(id)"	NA	NA
	employee_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (employee_id) REFERENCES employee(id)"	NA	
customer_insurance	customer_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (customer_id) REFERENCES customer(id)"	NA	NA
	insurance_id	RAW(16)	PRIMARY KEY, NOT NULL, FOREIGN KEY (insurance_id) REFERENCES insurance(id)"	NA	
	status	CHAR(8)	NOT NULL	NA	

My Contributions to Project:

Please find my contributions to build the project description for a database to a AutoZone, a National retailer of automotive parts and accessories Company.

- I have designed the following entities along with their attributes and relations.
    - Automotive Retailer - The "Automotive Retailer" is a major element in our database system, representing businesses engaged in the automotive retail industry. All the attributes in Automotive retailer collectively define and manage essential information about the retailer. Its relationships with employees, inventory, addresses, and service jobs enhance efficiency and effectiveness in serving customers within the automotive retail industry.
    - Inventory Product -The "Inventory Product" plays a pivotal role in the system, serving as a comprehensive repository for managing various automotive products. It will define and track vital information about each product. Its relationships with suppliers, retailers, billing records, service jobs, automobiles, and addresses facilitate comprehensive inventory management and effective product tracking within the system.
    - Automobile –The "Automobile" serves as a fundamental component for managing vehicle-related data available in the automotive industry. It ensures comprehensive information about each automobile. Its relationships with inventory products and service jobs facilitate efficient tracking and management of automotive inventory and service history within the system.
    - Supplier – The "Supplier" is for managing the sourcing of inventory products within the system. Its relationships with products and addresses enhance the efficiency and accuracy of supply chain management and communication with suppliers.
- I have led this team of 4 and assisted them in
    - Distribution of work in the project.
    - Guiding to design the entities, attributes, and their relations.
    - Consolidation of the project.