American International University -Bangladesh

Final Term Report

Project Title: Cupcake Shop Management System

Course Name: ADVANCE DATABASE MANAGEMENT SYSTEM

Course Teacher: Rezwan Ahmed

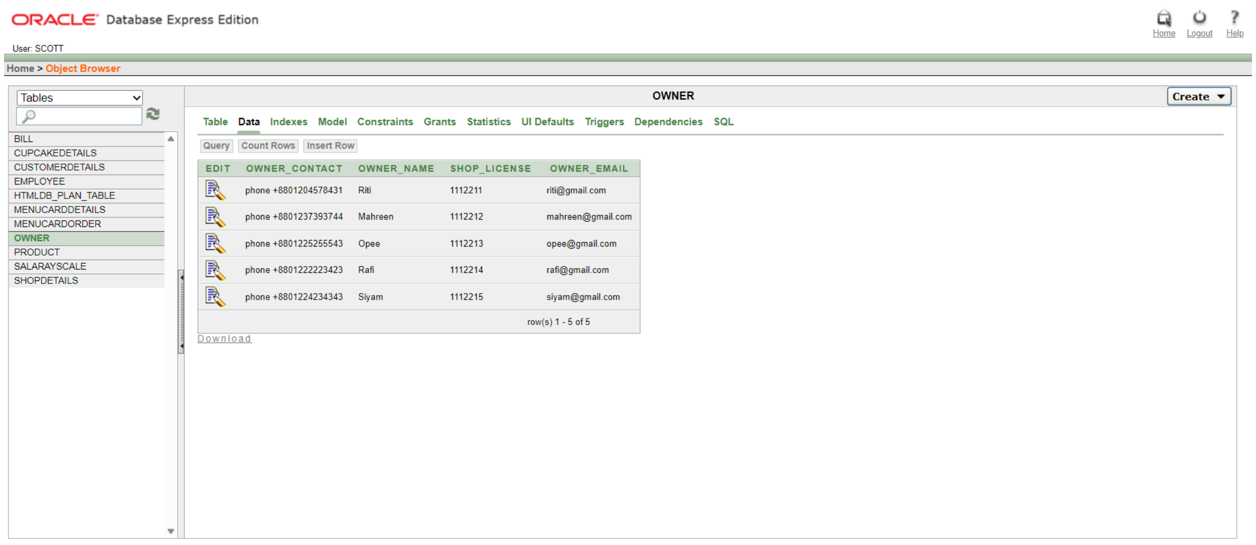
Semester: Spring 23-24

NAME	ID	SECTION
RANA TABASSUM	20-42124-1	C
SHAFIUL AJAM OPEE	20-43194-1	C
MAHREEN TABASSUM	20-43306-1	C

TABLE OF CONTENTS

Oracle Database

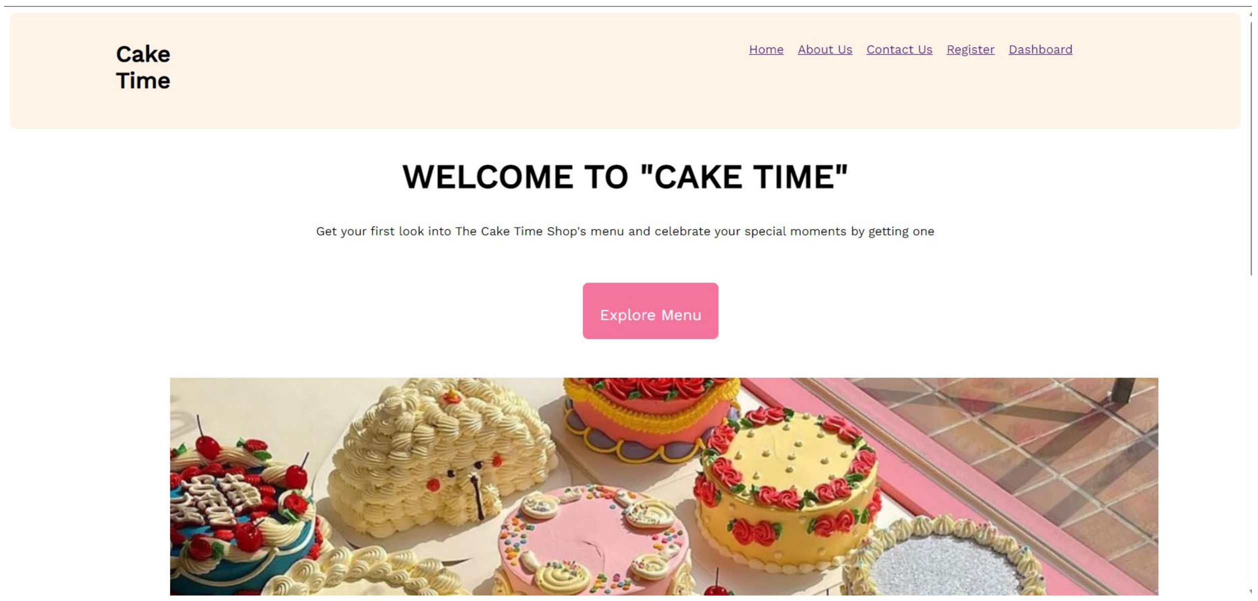
Here we create an admin portal and for storing admin information we created this database 'Admin' in oracle.



EDIT	OWNER_CONTACT	OWNER_NAME	SHOP_LICENSE	OWNER_EMAIL
	phone +8801204578431	Riti	1112211	riti@gmail.com
	phone +8801237393744	Mahreen	1112212	mahreen@gmail.com
	phone +8801225255543	Opee	1112213	opee@gmail.com
	phone +8801222223423	Rafi	1112214	rafi@gmail.com
	phone +8801224234343	Siyam	1112215	siyam@gmail.com

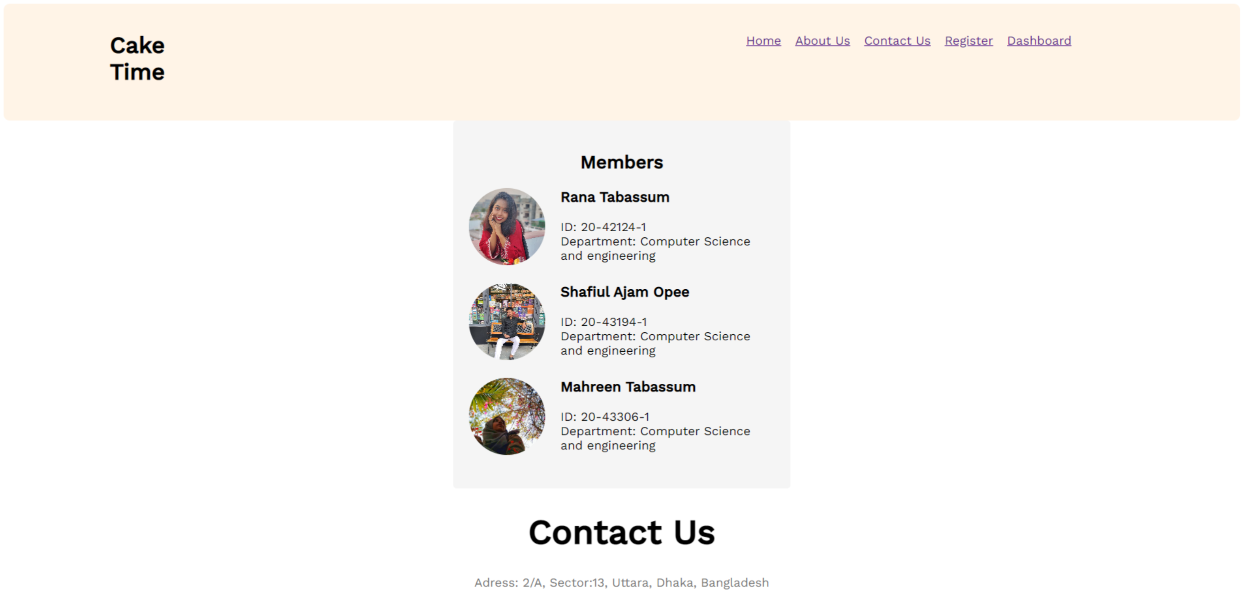
row(s) 1 - 5 of 5

User Interface



WELCOME TO "CAKE TIME"

Get your first look into The Cake Time Shop's menu and celebrate your special moments by getting one



Cake Time

Home About Us Contact Us Register Dashboard

Members

Rana Tabassum

ID: 20-42124-1
Department: Computer Science and engineering

Shafiul Ajam Opee

ID: 20-43194-1
Department: Computer Science and engineering

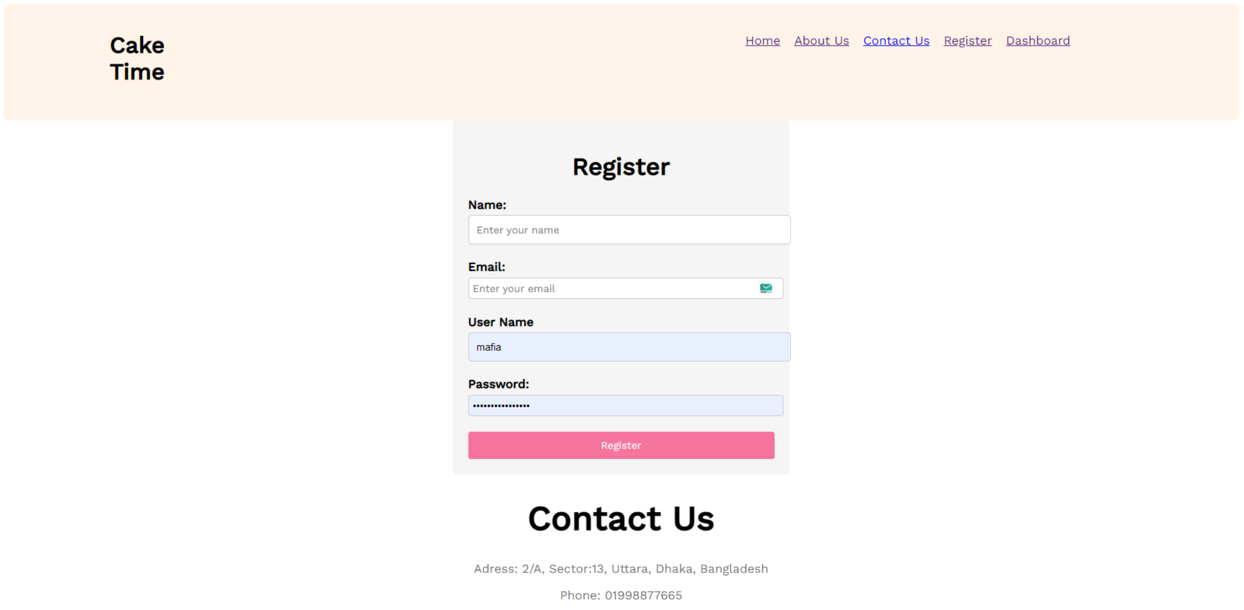
Mahreen Tabassum

ID: 20-43306-1
Department: Computer Science and engineering

Contact Us

Adress: 2/A, Sector:13, Uttara, Dhaka, Bangladesh

Here is our registration page.



Cake Time

Home About Us Contact Us Register Dashboard

Register

Name: Enter your name

Email: Enter your email

User Name: mafia

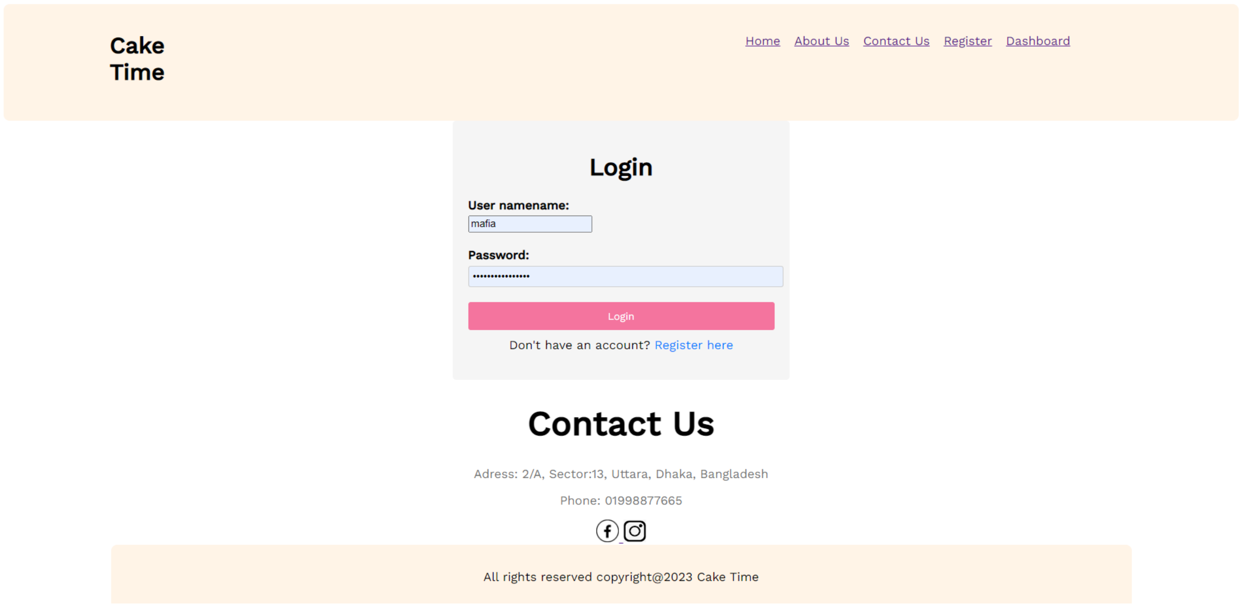
Password: ••••••••••••••••

Contact Us

Adress: 2/A, Sector:13, Uttara, Dhaka, Bangladesh

Phone: 01998877665

After register we have to login through this interface.


Cake Time

Home About Us Contact Us Register Dashboard

Login

User namename: mafia

Password: ••••••••••••••••

Login

Don't have an account? Register here

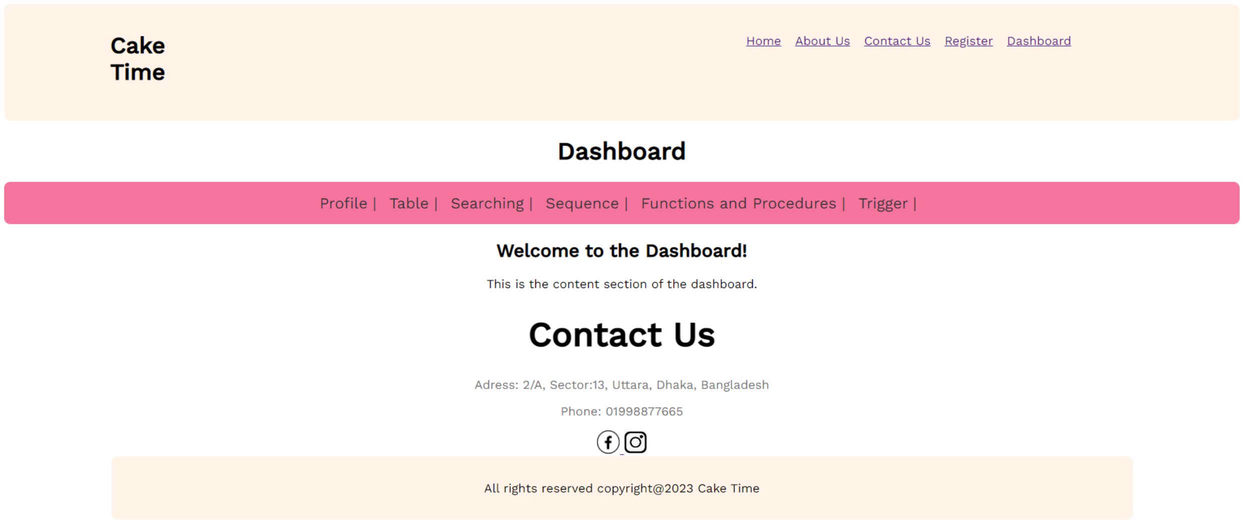
Contact Us

Adress: 2/A, Sector:13, Uttara, Dhaka, Bangladesh

Phone: 01998877665

All rights reserved copyright@2023 Cake Time

After giving correct username and password we entered to Dashboard.


Cake Time

Home About Us Contact Us Register Dashboard

Dashboard

Profile | Table | Searching | Sequence | Functions and Procedures | Trigger |

Welcome to the Dashboard!

This is the content section of the dashboard.

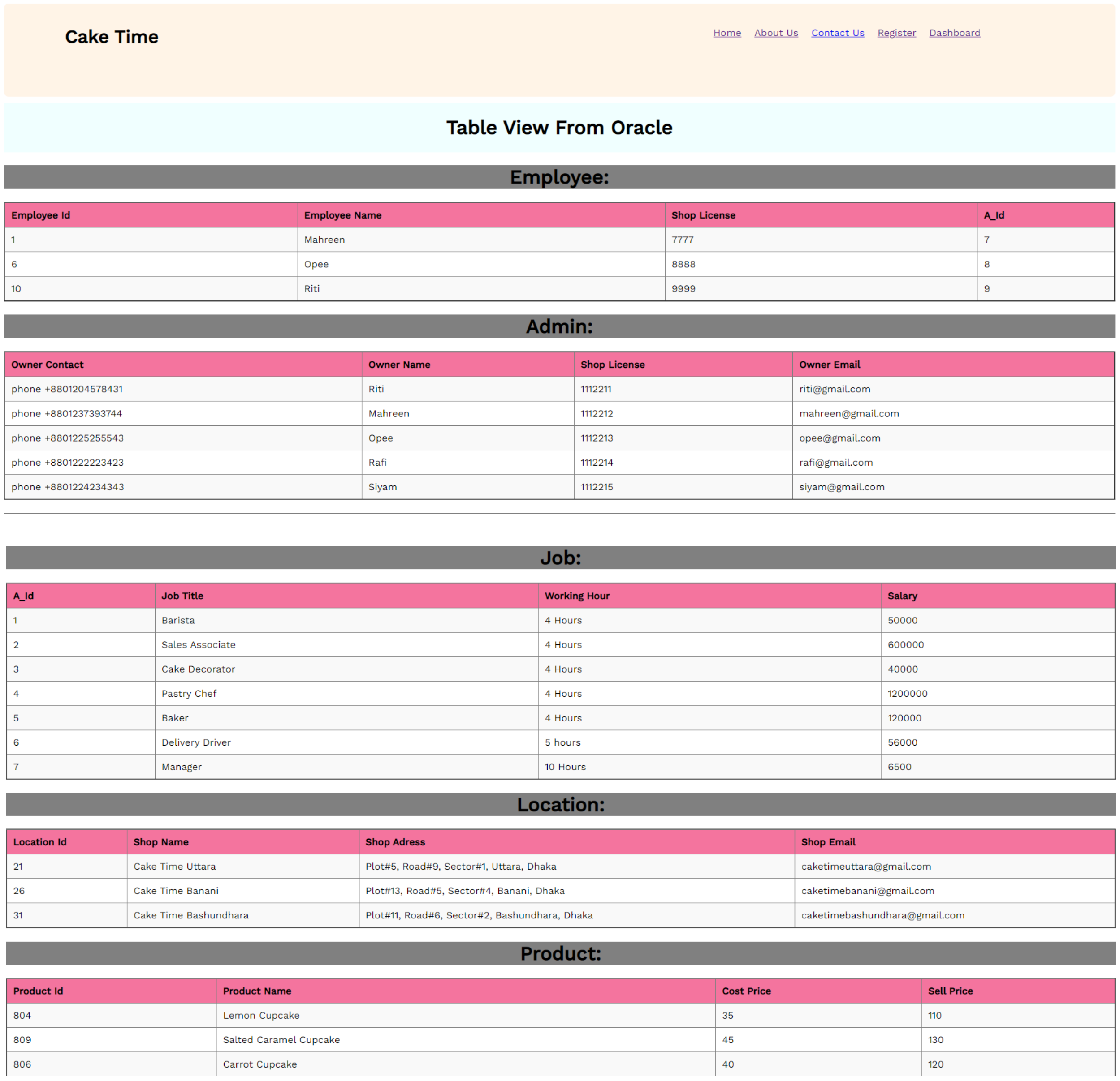
Contact Us

Adress: 2/A, Sector:13, Uttara, Dhaka, Bangladesh

Phone: 01998877665

All rights reserved copyright@2023 Cake Time

Now we will show this data in admin panel through view.



Cake Time

Home About Us Contact Us Register Dashboard

Table View From Oracle

Employee:

Employee Id	Employee Name	Shop License	A_Id
1	Mahreen	7777	7
6	Opee	8888	8
10	Riti	9999	9

Admin:

Owner Contact	Owner Name	Shop License	Owner Email
phone +8801204578431	Riti	1112211	riti@gmail.com
phone +8801237393744	Mahreen	1112212	mahreen@gmail.com
phone +8801225255543	Opee	1112213	opee@gmail.com
phone +8801222223423	Rafi	1112214	rafi@gmail.com
phone +8801224234343	Siyam	1112215	siyam@gmail.com

Job:

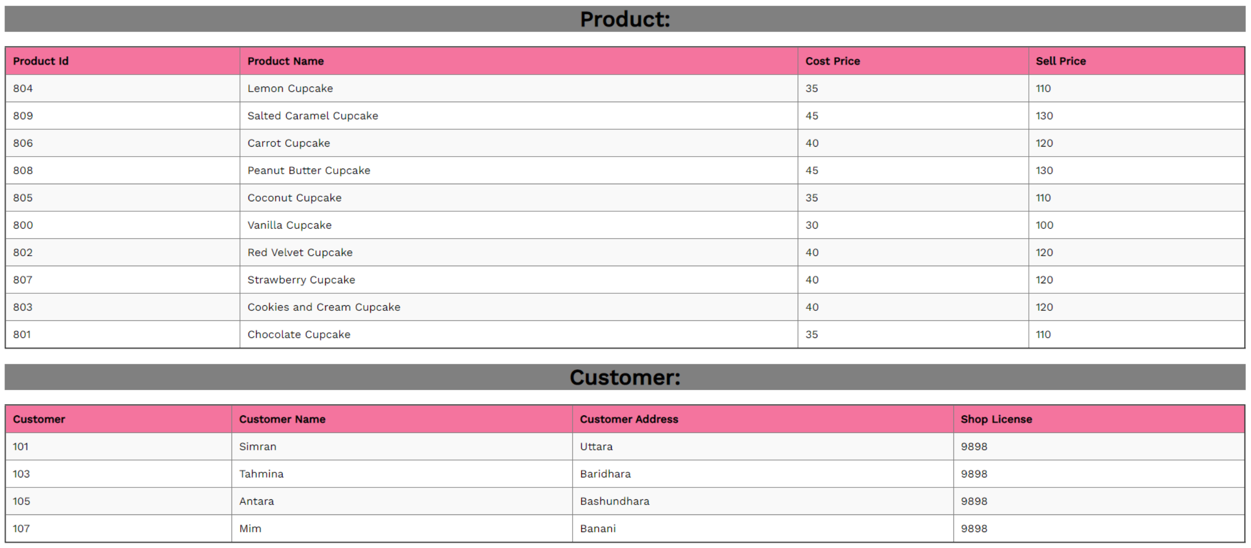
A_Id	Job Title	Working Hour	Salary
1	Barista	4 Hours	50000
2	Sales Associate	4 Hours	600000
3	Cake Decorator	4 Hours	40000
4	Pastry Chef	4 Hours	1200000
5	Baker	4 Hours	120000
6	Delivery Driver	5 hours	56000
7	Manager	10 Hours	6500

Location:

Location Id	Shop Name	Shop Adress	Shop Email
21	Cake Time Uttara	Plot#5, Road#9, Sector#1, Uttara, Dhaka	caketimeuttara@gmail.com
26	Cake Time Banani	Plot#13, Road#5, Sector#4, Banani, Dhaka	caketimebanani@gmail.com
31	Cake Time Bashundhara	Plot#11, Road#6, Sector#2, Bashundhara, Dhaka	caketimebashundhara@gmail.com

Product:

Product Id	Product Name	Cost Price	Sell Price
804	Lemon Cupcake	35	110
809	Salted Caramel Cupcake	45	130
806	Carrot Cupcake	40	120



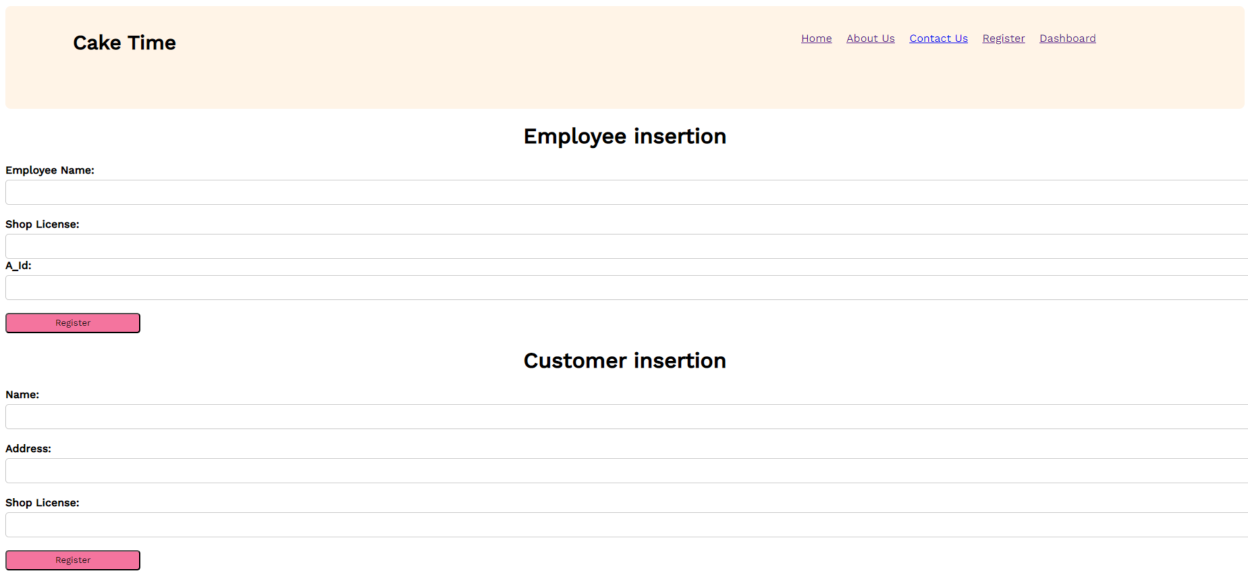
Product:

Product Id	Product Name	Cost Price	Sell Price
804	Lemon Cupcake	35	110
809	Salted Caramel Cupcake	45	130
806	Carrot Cupcake	40	120
808	Peanut Butter Cupcake	45	130
805	Coconut Cupcake	35	110
800	Vanilla Cupcake	30	100
802	Red Velvet Cupcake	40	120
807	Strawberry Cupcake	40	120
803	Cookies and Cream Cupcake	40	120
801	Chocolate Cupcake	35	110

Customer:

Customer	Customer Name	Customer Address	Shop License
101	Simran	Uttara	9898
103	Tahmina	Baridhara	9898
105	Antara	Bashundhara	9898
107	Mim	Banani	9898

Here We have implemented sequence. We didn't give insert id for employee and customer table as it will be updated automatically through sequence.



Employee insertion

Employee Name:

Shop License:

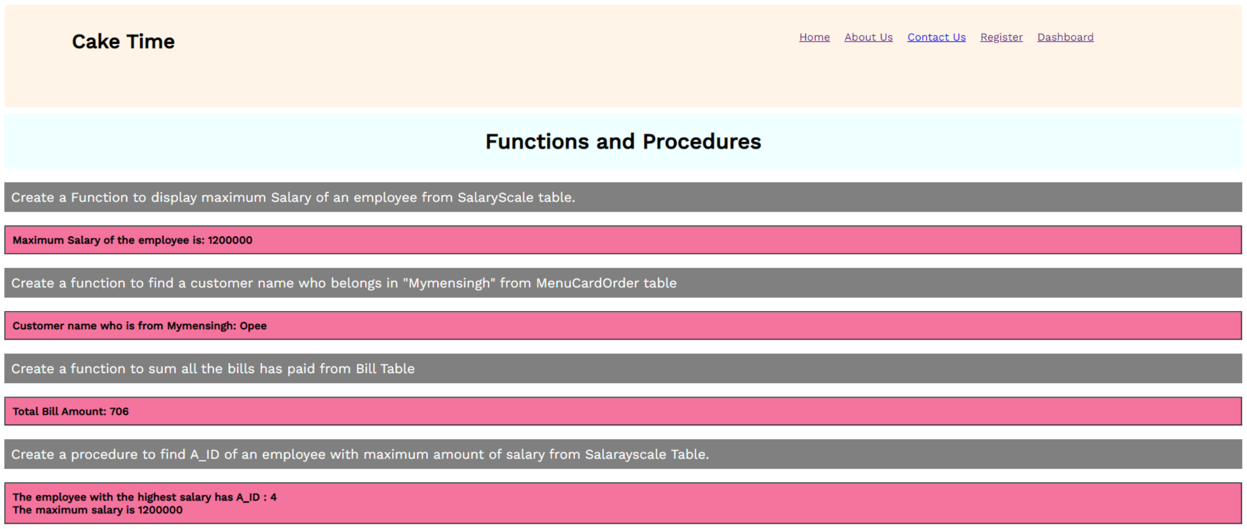
A_Id:

Customer insertion

Name:

Address:

Shop License:


Cake Time

Home About Us Contact Us Register Dashboard

Functions and Procedures

Create a Function to display maximum Salary of an employee from SalaryScale table.

Maximum Salary of the employee is: 1200000

Create a function to find a customer name who belongs in "Mymensingh" from MenuCardOrder table

Customer name who is from Mymensingh: Opee

Create a function to sum all the bills has paid from Bill Table

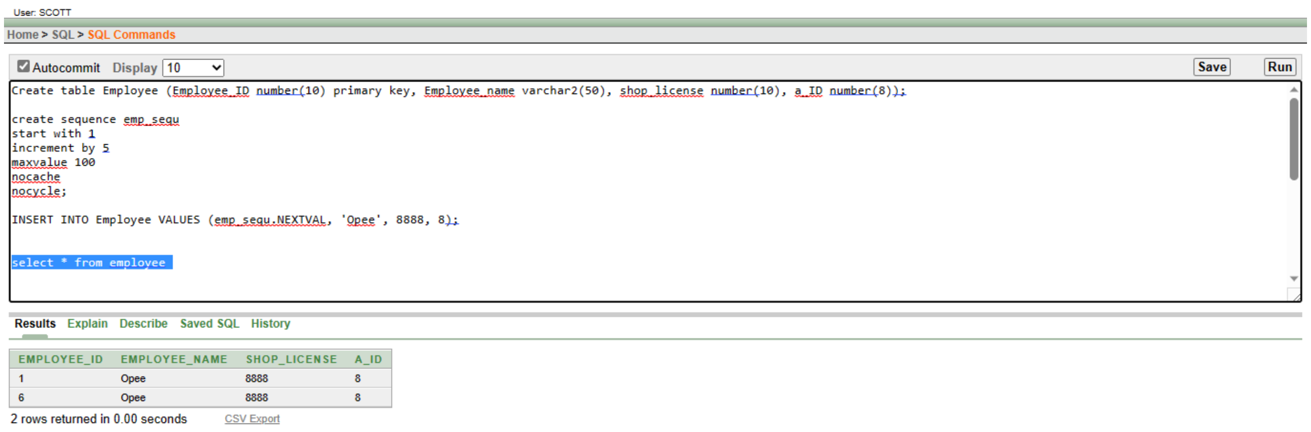
Total Bill Amount: 706

Create a procedure to find A_ID of an employee with maximum amount of salary from Salarayscale Table.

The employee with the highest salary has A_ID : 4
The maximum salary is 1200000

Sequence:

1. **As Employee_ID is a primary key it will be auto generated due to creation of a sequence under the table Employee.**


User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History
CSV Export

```
Create table Employee (Employee_ID number(10) primary key, Employee_name varchar2(50), shop_license number(10), a_ID number(8));

create sequence emp_sequ
start with 1
increment by 5
maxvalue 100
nocache
nocycle;

INSERT INTO Employee VALUES (emp_sequ.NEXTVAL, 'Opee', 8888, 8);


select * from employee
```

EMPLOYEE_ID	EMPLOYEE_NAME	SHOP_LICENSE	A_ID
1	Opee	8888	8
6	Opee	8888	8

2 rows returned in 0.00 seconds

Create table Employee (Employee_ID number(10) primary key, Employee_name varchar2(50), shop_license number(10), a_ID number(8));

create sequence emp_sequ

start with 1

increment by 5

maxvalue 100

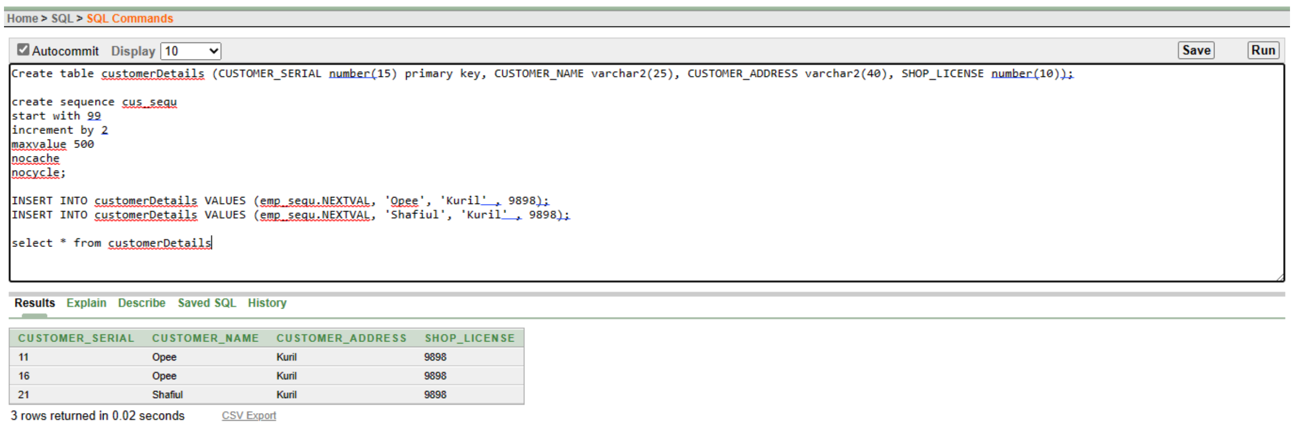
nocache

nocycle;

INSERT INTO Employee VALUES (emp_sequ.NEXTVAL, 'Opee', 8888, 8);

select * from employee.

2. **As CUSTOMER_SERIAL is a primary key it will be auto generated due to creation of a sequence called cus_sequ under the table customerDetails.**



```
Create table customerDetails (CUSTOMER_SERIAL number(15) primary key, CUSTOMER_NAME varchar2(25), CUSTOMER_ADDRESS varchar2(40), SHOP_LICENSE number(10));

create sequence cus_sequ
start with 99
increment by 2
maxvalue 500
nocache
nocycle;

INSERT INTO customerDetails VALUES (emp_sequ.NEXTVAL, 'Opee', 'Kuril' , 9898);
INSERT INTO customerDetails VALUES (emp_sequ.NEXTVAL, 'Shafiul', 'Kuril' , 9898);

select * from customerDetails
```

CUSTOMER_SERIAL	CUSTOMER_NAME	CUSTOMER_ADDRESS	SHOP_LICENSE
11	Opee	Kuril	9898
16	Opee	Kuril	9898
21	Shafiul	Kuril	9898

3 rows returned in 0.02 seconds

Create table customerDetails (

CUSTOMER_SERIAL number(15) primary key,

CUSTOMER_NAME varchar2(25),

CUSTOMER_ADDRESS varchar2(40),

SHOP_LICENSE number(10));

create sequence cus_sequ

start with 99

increment by 2

maxvalue 500

nocache

nocycle;

INSERT INTO customerDetails VALUES (cus_sequ.NEXTVAL, 'Opee', 'Kuril' , 9898);

INSERT INTO customerDetails VALUES (cus_sequ.NEXTVAL, 'Shafiul', 'Kuril' , 9898);

select * from customerDetails

3. As shop_license is a primary key it will be auto generated due to creation of a sequence called shop_sequ under the table shopDetails.

Create table shopDetails (shop_license number(10) primary key, Shop_Name varchar2(40),

Shop_address varchar2(90), shop_Email varchar2(100));

create sequence shop_sequ

start with 25

increment by 2

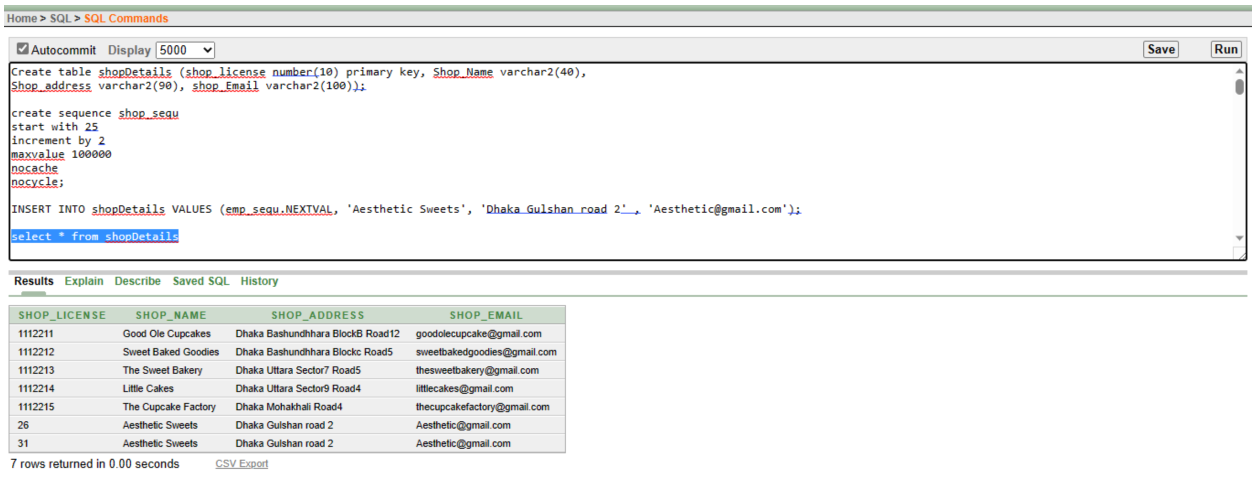
maxvalue 100000

nocache

nocycle;

INSERT INTO shopDetails VALUES (shop_sequ.NEXTVAL, 'Aesthetic Sweets', 'Dhaka Gulshan road 2' , 'Aesthetic@gmail.com');

select * from shopDetails


Home > SQL > SQL Commands
Autocommit Display 5000
Save
Run
Results Explain Describe Saved SQL History

```
Create table shopDetails (shop_license number(10) primary key, Shop_Name varchar2(40),
Shop_address varchar2(90), shop_Email varchar2(100));

create sequence shop_sequ
start with 25
increment by 2
maxvalue 100000
nocache
nocycle;

INSERT INTO shopDetails VALUES (emp_sequ.NEXTVAL, 'Aesthetic Sweets', 'Dhaka Gulshan road 2' , 'Aesthetic@gmail.com');

select * from shopDetails
```

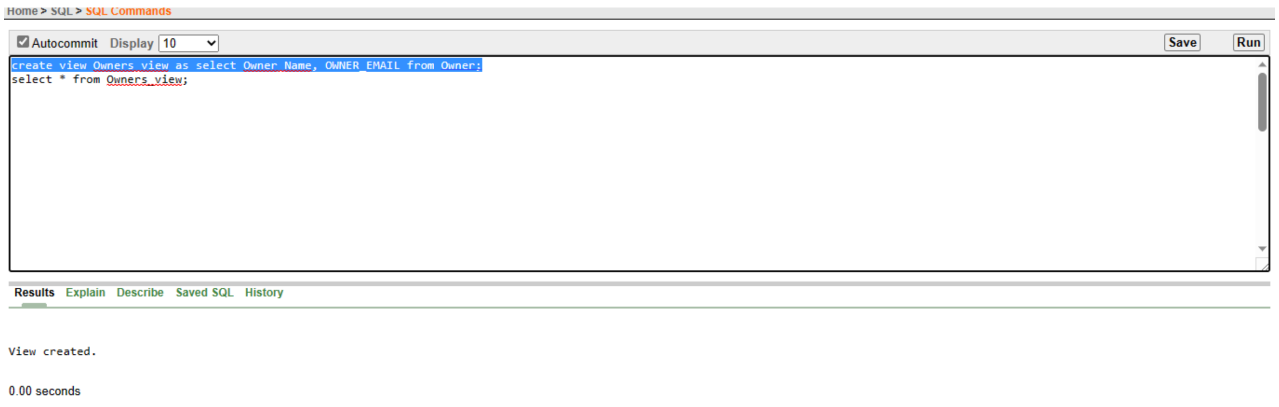
SHOP_LICENSE	SHOP_NAME	SHOP_ADDRESS	SHOP_EMAIL
1112211	Good Ole Cupcakes	Dhaka Bashundhhara BlockB Road12	goodolecupcake@gmail.com
1112212	Sweet Baked Goodies	Dhaka Bashundhhara Blockc Road5	sweetbakedgoodies@gmail.com
1112213	The Sweet Bakery	Dhaka Uttara Sector7 Road5	thesweetbakery@gmail.com
1112214	Little Cakes	Dhaka Uttara Sector9 Road4	littlecakes@gmail.com
1112215	The Cupcake Factory	Dhaka Mohakhali Road4	thecupcakefactory@gmail.com
26	Aesthetic Sweets	Dhaka Gulshan road 2	Aesthetic@gmail.com
31	Aesthetic Sweets	Dhaka Gulshan road 2	Aesthetic@gmail.com

7 rows returned in 0.00 seconds   CSV Export

Views:

01. Create a view called Owners_VIEW based on Owner_Name and OWNER_EMAIL from the Owner Table

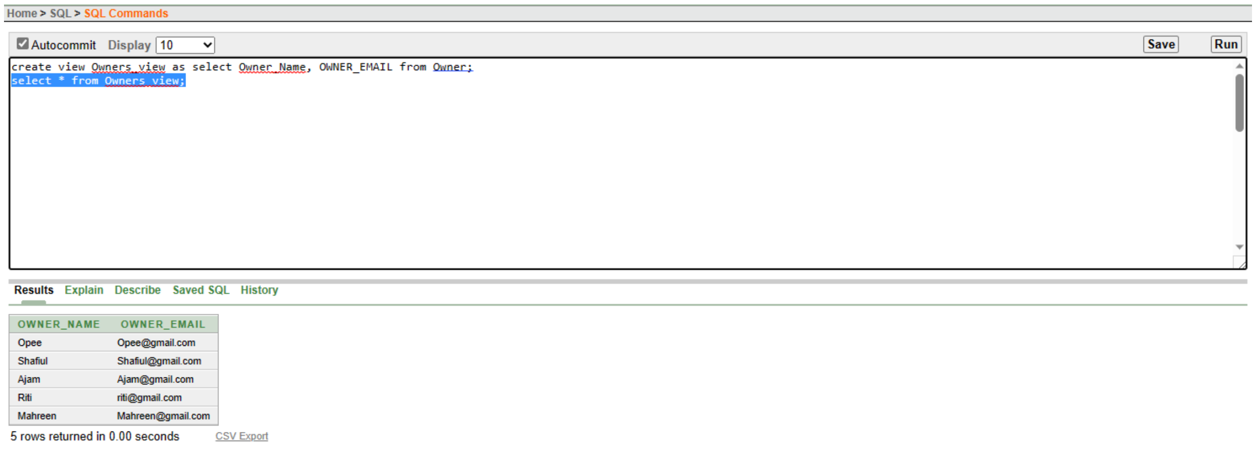
create view Owners_view
as
select Owner_Name, OWNER_EMAIL from Owner;
select * from Owners_view;


Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

```
create view Owners_view as select Owner_Name, OWNER_EMAIL from Owner;
select * from Owners_view;
```

```
View created.
```

0.00 seconds


Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History
CSV Export

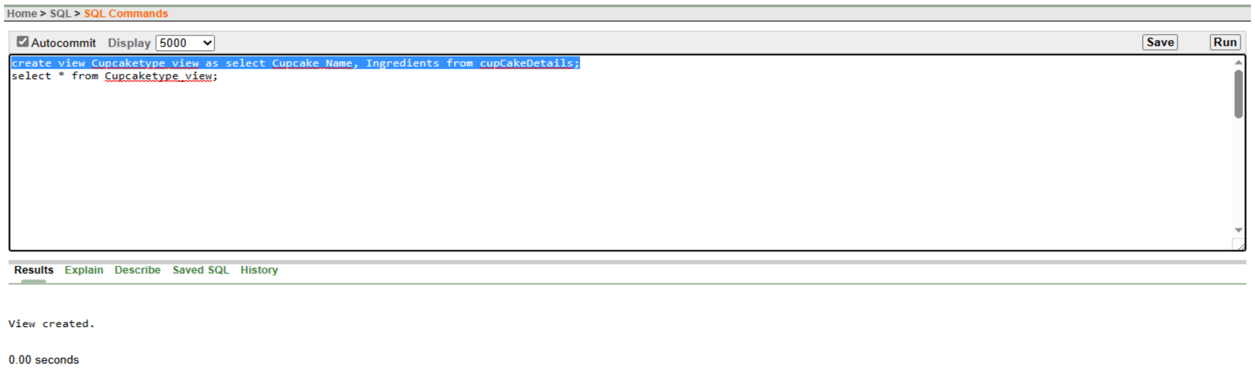
```
create view Owners_view as select Owner_Name, OWNER_EMAIL from Owner;
select * from Owners_view;
```

OWNER_NAME	OWNER_EMAIL
Opee	Opee@gmail.com
Shafiul	Shafiul@gmail.com
Ajam	Ajam@gmail.com
Riti	riti@gmail.com
Mahreen	Mahreen@gmail.com

5 rows returned in 0.00 seconds

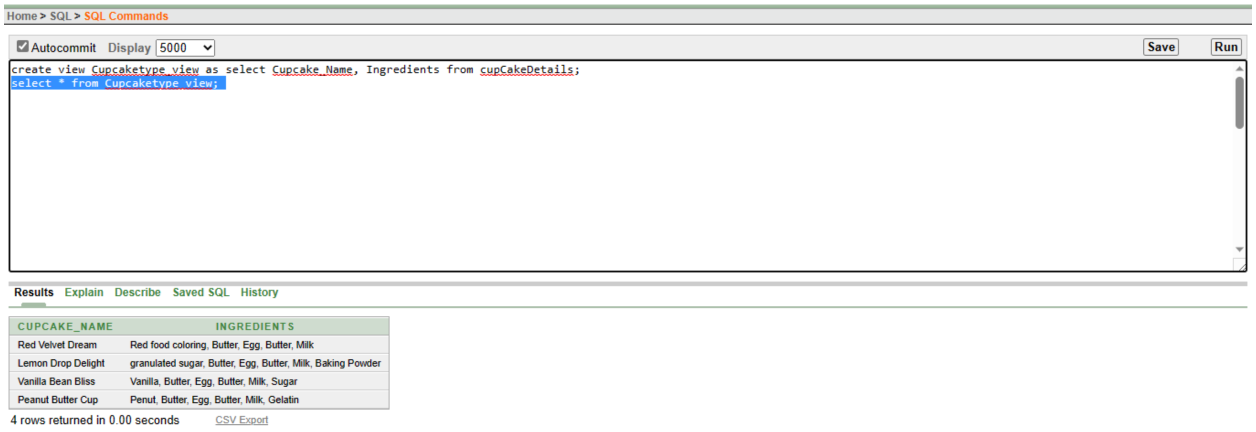
02. Create a view called Cupcaketype_view based on Cupcake_Name and Ingredients from the cupCakeDetailsTable

create view Cupcaketype_view as select Cupcake_Name, Ingredients from cupCakeDetails;
select * from Cupcaketype_view;


Home > SQL > SQL Commands
Autocommit Display 5000
Save
Run
Results Explain Describe Saved SQL History

```
create view Cupcaketype_view as select Cupcake_Name, Ingredients from cupCakeDetails;
select * from Cupcaketype_view;
```

```
View created.

0.00 seconds
```


Home > SQL > SQL Commands
Autocommit Display 5000
Save
Run
Results Explain Describe Saved SQL History
CSV Export

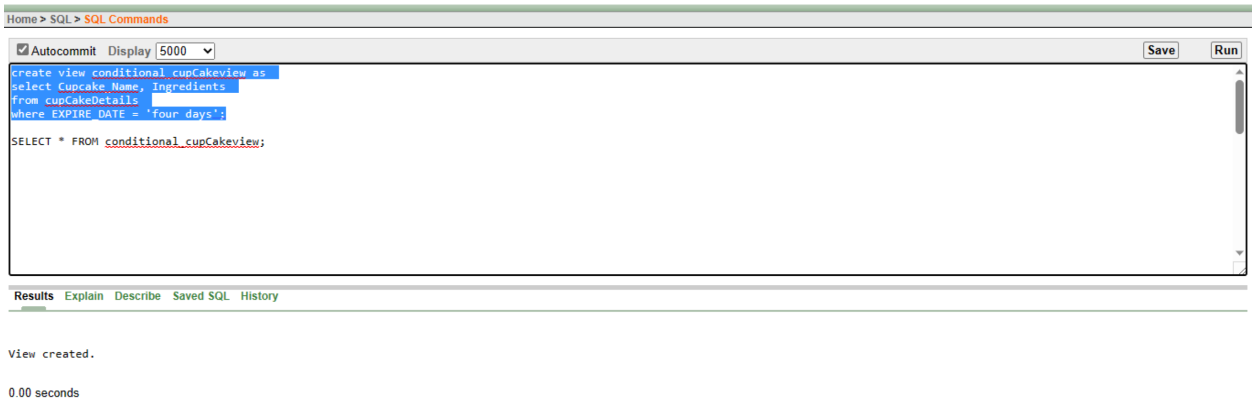
```
create view Cupcaketype_view as select Cupcake_Name, Ingredients from cupCakeDetails;
select * from Cupcaketype_view;
```

CUPCAKE_NAME	INGREDIENTS
Red Velvet Dream	Red food coloring, Butter, Egg, Butter, Milk
Lemon Drop Delight	granulated sugar, Butter, Egg, Butter, Milk, Baking Powder
Vanilla Bean Bliss	Vanilla, Butter, Egg, Butter, Milk, Sugar
Peanut Butter Cup	Penut, Butter, Egg, Butter, Milk, Gelatin

4 rows returned in 0.00 seconds

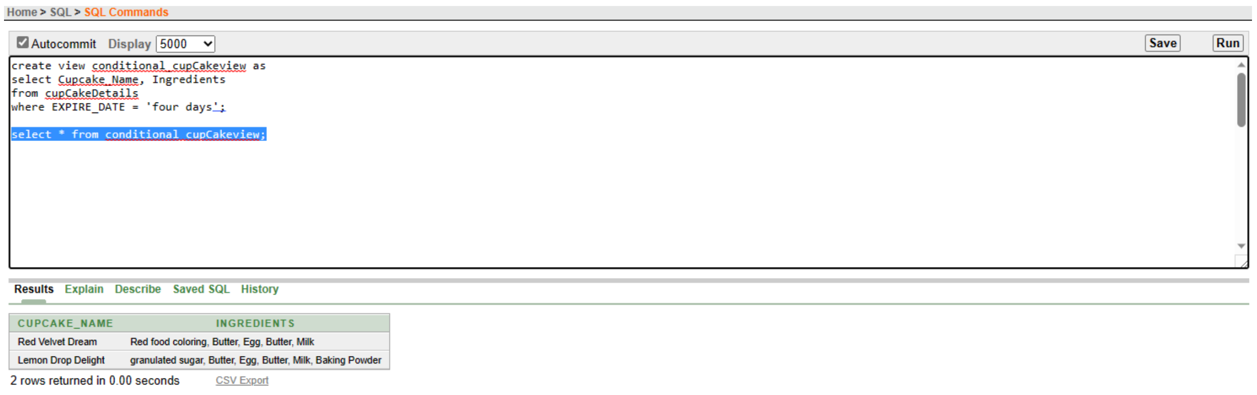
03. Create a view called conditional_cupCakeview to show the Cupcake_Name and Ingredients where cupcake is lasting 4 days from cupCakeDetails Table.

create view conditional_cupCakeview as
select Cupcake_Name, Ingredients
from cupCakeDetails s
where EXPIRE_DATE = 'four days';
select * from conditional_cupCakeview;


Home > SQL > SQL Commands
Autocommit Display 5000
Save
Run
Results Explain Describe Saved SQL History

```
create view conditional_cupCakeview as
select Cupcake_Name, Ingredients
from cupCakeDetails
where EXPIRE_DATE = 'four days';

SELECT * FROM conditional_cupCakeview;
```

```
View created.

0.00 seconds
```


Home > SQL > SQL Commands

Autocommit Display 5000 Save Run

```
create view conditional_cupCakeview as
select Cupcake_Name, Ingredients
from cupCakeDetails
where EXPIRE_DATE = 'four days';

select * from conditional_cupCakeview;
```

Results Explain Describe Saved SQL History

CUPCAKE_NAME	INGREDIENTS
Red Velvet Dream	Red food coloring, Butter, Egg, Butter, Milk
Lemon Drop Delight	granulated sugar, Butter, Egg, Butter, Milk, Baking Powder

2 rows returned in 0.00 seconds CSV Export

04. Create a view called maxavgsal_view to select A_ID of Employees from SalaryScale where their salary is more than average employee salary?

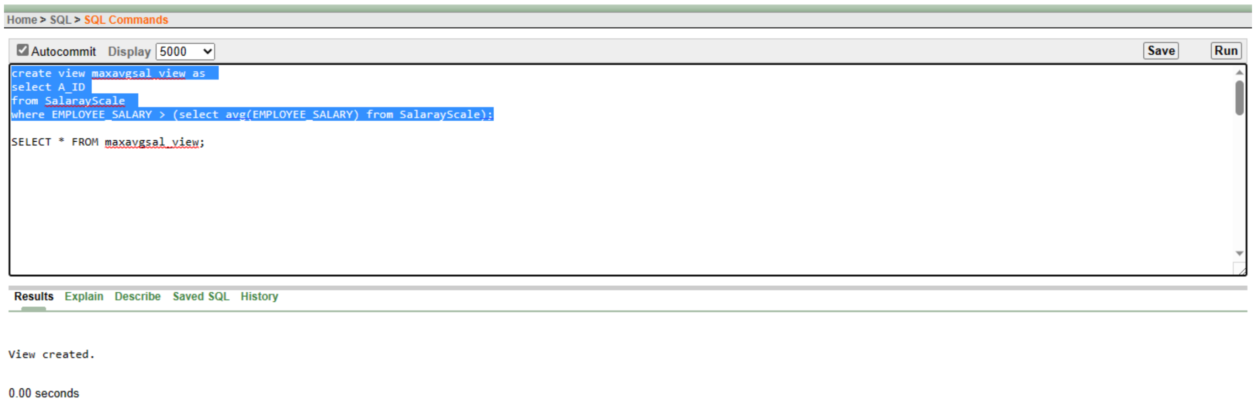
create view maxavgsal_view as

select A_ID

from SalarayScale

where EMPLOYEE_SALARY > (select avg(EMPLOYEE_SALARY) from SalarayScale);

SELECT * FROM maxavgsal_view;


Home > SQL > SQL Commands

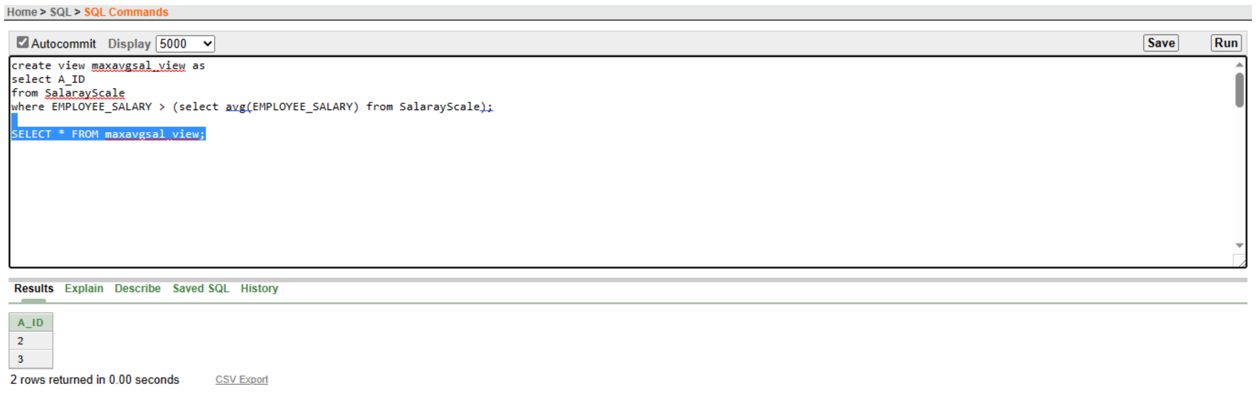
Autocommit Display 5000 Save Run

```
create view maxavgsal_view as
select A_ID
from SalarayScale
where EMPLOYEE_SALARY > (select avg(EMPLOYEE_SALARY) from SalarayScale);

SELECT * FROM maxavgsal_view;
```

Results Explain Describe Saved SQL History

```
View created.
```

0.00 seconds


Home > SQL > SQL Commands

Autocommit Display 5000 Save Run

```
create view maxavgsal_view as
select A_ID
from SalarayScale
where EMPLOYEE_SALARY > (select avg(EMPLOYEE_SALARY) from SalarayScale);

SELECT * FROM maxavgsal_view;
```

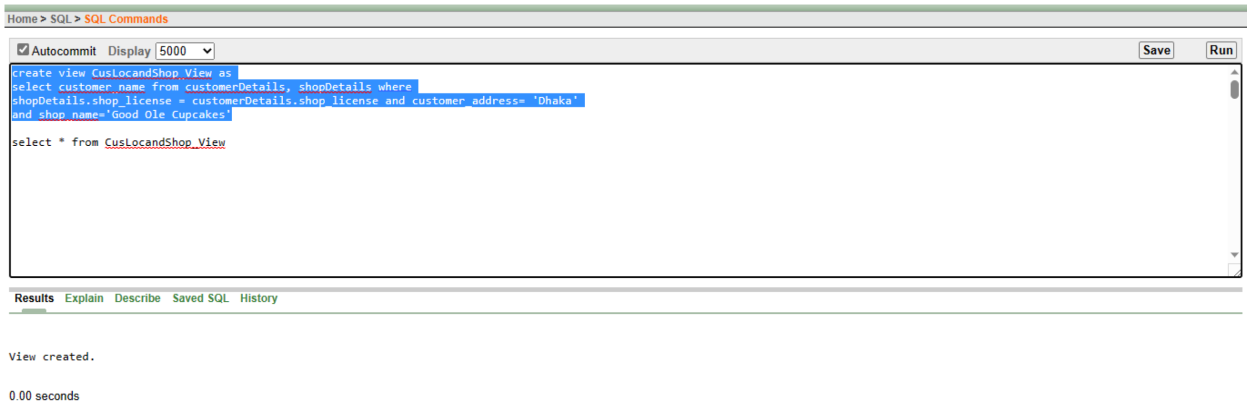
Results Explain Describe Saved SQL History

A_ID
2
3

2 rows returned in 0.00 seconds CSV Export

05. Create a view called CusLocandShop _View to find the customer name who belongs to Dhaka and goes to the Good Ole Cupcakes

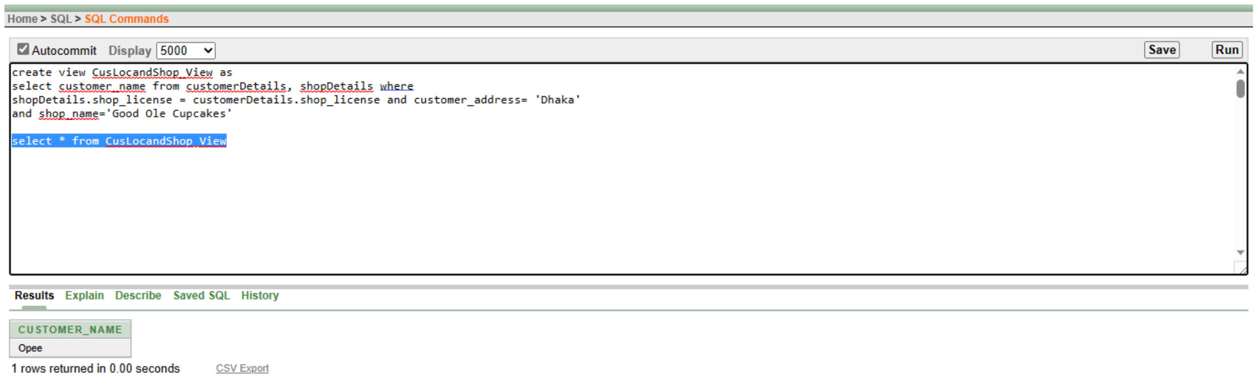
```
create view CusLocandShop_View as
select customer_name from customerDetails, shopDetails where
shopDetails.shop_license = customerDetails.shop_license and customer_address= 'Dhaka'
and shop_name='Good Ole Cupcakes'

select * from CusLocandShop_View
```


Home > SQL > SQL Commands
Autocommit Display 5000
Save
Run
Results Explain Describe Saved SQL History

```
create view CusLocandShop_View as
select customer_name from customerDetails, shopDetails where
shopDetails.shop_license = customerDetails.shop_license and customer_address= 'Dhaka'
and shop_name='Good Ole Cupcakes'

select * from CusLocandShop_View
```

```
View created.
```

0.00 seconds


Home > SQL > SQL Commands
Autocommit Display 5000
Save
Run
Results Explain Describe Saved SQL History
CSV Export

```
create view CusLocandShop_View as
select customer_name from customerDetails, shopDetails where
shopDetails.shop_license = customerDetails.shop_license and customer_address= 'Dhaka'
and shop_name='Good Ole Cupcakes'

select * from CusLocandShop_View
```

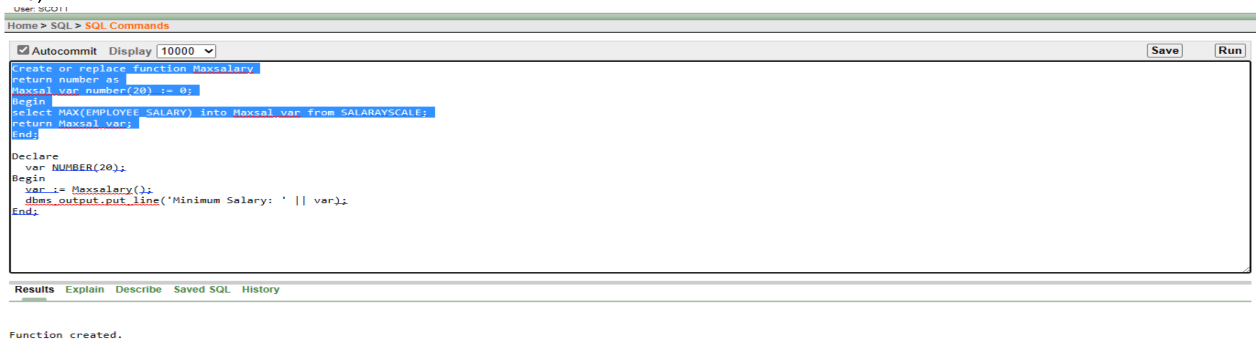
CUSTOMER_NAME
Opee

1 rows returned in 0.00 seconds

Function

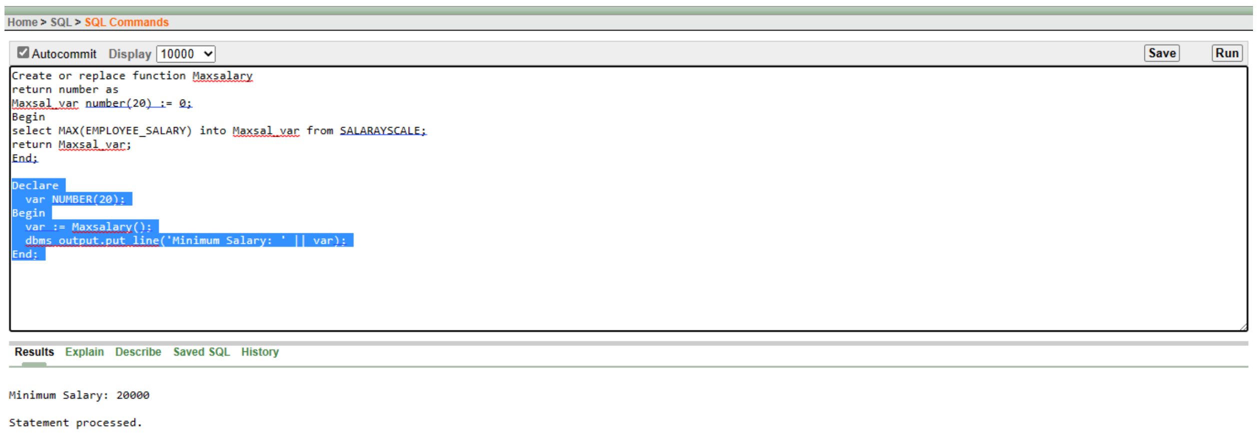
1. Create a Function to display maximum Salary of an employee from SalaryScale table.

```
Create or replace function Maxsalary
return number as
Maxsal_var number(20) := 0;
Begin
select MAX(EMPLOYEE_SALARY) into Maxsal_var from SALARAYSCALE;
return Maxsal_var;
End;

Declare
  var NUMBER(20);
Begin
  var := Maxsalary();
  dbms_output.put_line('Maximum Salary: ' || var);
End;
```


User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 10000
Save
Run
Results Explain Describe Saved SQL History

```
Create or replace function Maxsalary
return number as
Maxsal_var number(20) := 0;
Begin
select MAX(EMPLOYEE_SALARY) into Maxsal_var from SALARAYSCALE;
return Maxsal_var;
End;

Declare
  var NUMBER(20);
Begin
  var := Maxsalary();
  dbms_output.put_line('Minimum Salary: ' || var);
End;
```

```
Function created.
```


Home > SQL > SQL Commands
Autocommit Display 10000
Save
Run
Results Explain Describe Saved SQL History

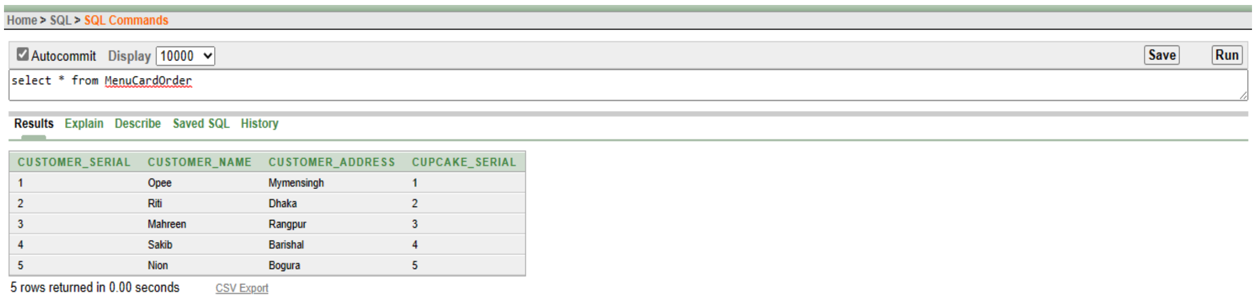
```
Create or replace function Maxsalary
return number as
Maxsal_var number(20) := 0;
Begin
select MAX(EMPLOYEE_SALARY) into Maxsal_var from SALARAYSCALE;
return Maxsal_var;
End;

Declare
  var NUMBER(20);
Begin
  var := Maxsalary();
  dbms_output.put_line('Minimum Salary: ' || var);
End;
```

```
Minimum Salary: 20000

Statement processed.
```

2. Create a function to find a customer name who belongs in ‘Mymensingh’ from MenuCardOrder table

```
Create or replace function FindCustomerName(address in varchar2)
return varchar2
IS
  name varchar2(20);
Begin
  Select customer_Name INTO name from MenuCardOrder
  where customer_address = address;
  RETURN name;
End;

Declare
  name varchar2(20);
```

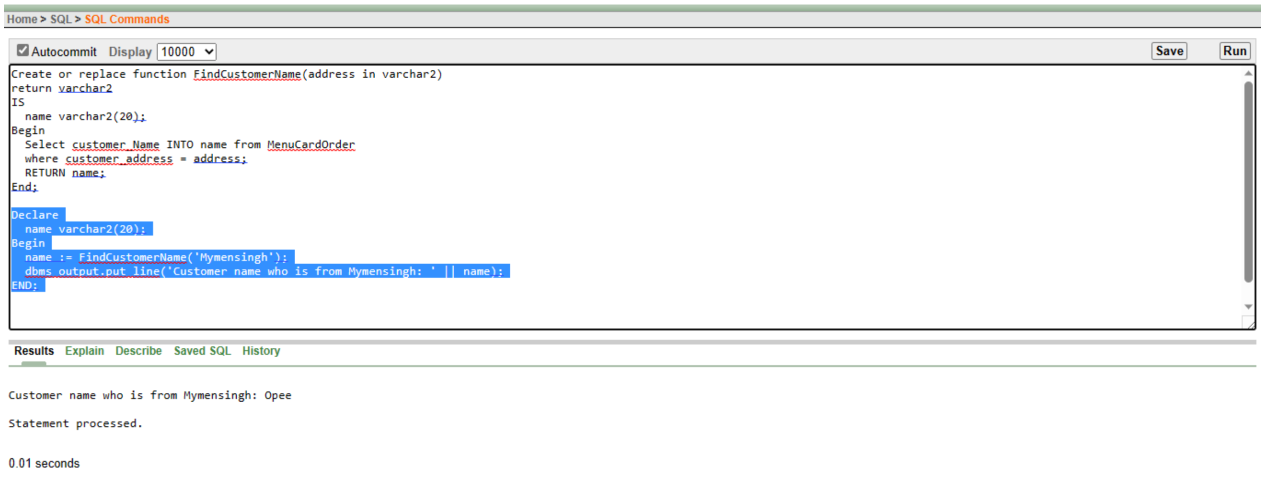
```
Begin
  name := FindCustomerName(' Rangpur ');
  dbms_output.put_line('Customer name who is from Rangpur: ' || name);
END;
```


Home > SQL > SQL Commands
Autocommit Display 10000
Save
Run
Results Explain Describe Saved SQL History
CSV Export

```
select * from MenuCardOrder
```

CUSTOMER_SERIAL	CUSTOMER_NAME	CUSTOMER_ADDRESS	CUPCAKE_SERIAL
1	Opee	Mymensingh	1
2	Riti	Dhaka	2
3	Mahreen	Rangpur	3
4	Sakib	Barishal	4
5	Nion	Bogura	5

5 rows returned in 0.00 seconds


Home > SQL > SQL Commands
Autocommit Display 10000
Save
Run
Results Explain Describe Saved SQL History

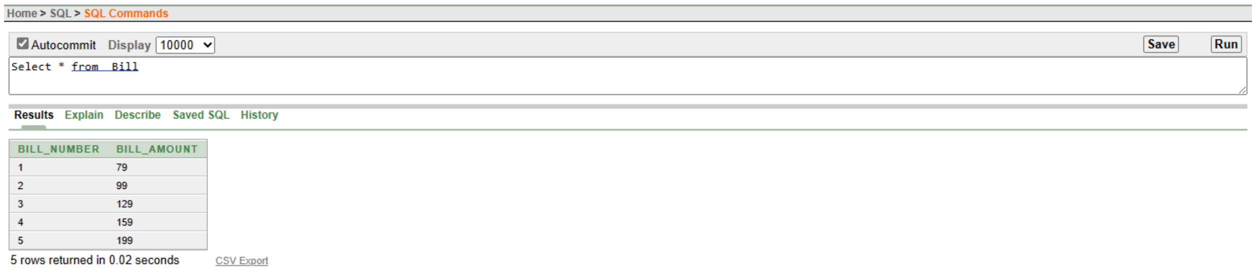
```
Create or replace function FindCustomerName(address in varchar2)
return varchar2
IS
  name varchar2(20);
Begin
  Select customer_Name INTO name from MenuCardOrder
  where customer_address = address;
  RETURN name;
End;

Declare
  name varchar2(20);
Begin
  name := FindCustomerName('Mymensingh');
  dbms_output.put_line('Customer name who is from Mymensingh: ' || name);
END;
```

```
Customer name who is from Mymensingh: Opee

Statement processed.
```

0.01 seconds

3. Create a function to sum all the bills has paid from Bill Table

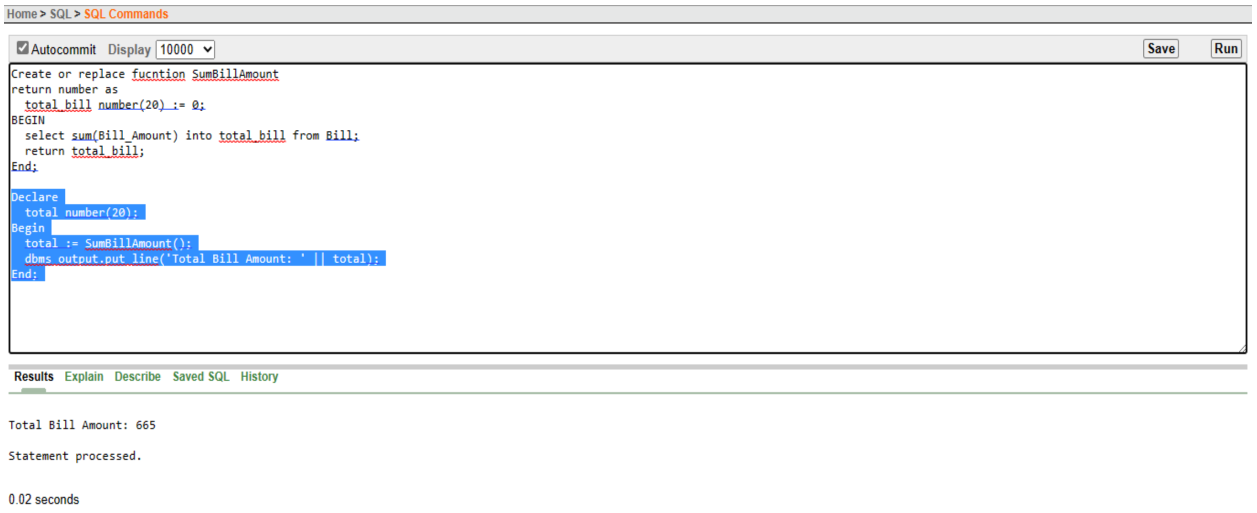
```
Create or replace function SumBillAmount
return number as
  total_bill number(20) := 0;
BEGIN
  select sum(Bill_Amount) into total_bill from Bill;
  return total_bill;
End;

Declare
  total number(20);
Begin
  total := SumBillAmount();
  dbms_output.put_line('Total Bill Amount: ' || total);
End;
```


Home > SQL > SQL Commands
Autocommit Display 10000
Save
Run
Results Explain Describe Saved SQL History
CSV Export

```
Select * from  Bill
```

BILL_NUMBER	BILL_AMOUNT
1	79
2	99
3	129
4	159
5	199

5 rows returned in 0.02 seconds


Home > SQL > SQL Commands
Autocommit Display 10000
Save
Run
Results Explain Describe Saved SQL History

```
Create or replace fucntion SumBillAmount
return number as
  total_bill number(20) := 0;
BEGIN
  select sum(Bill_Amount) into total_bill from Bill;
  return total_bill;
End;

Declare
  total number(20);
Begin
  total := SumBillAmount();
  dbms_output.put_line('Total Bill Amount: ' || total);
End;
```

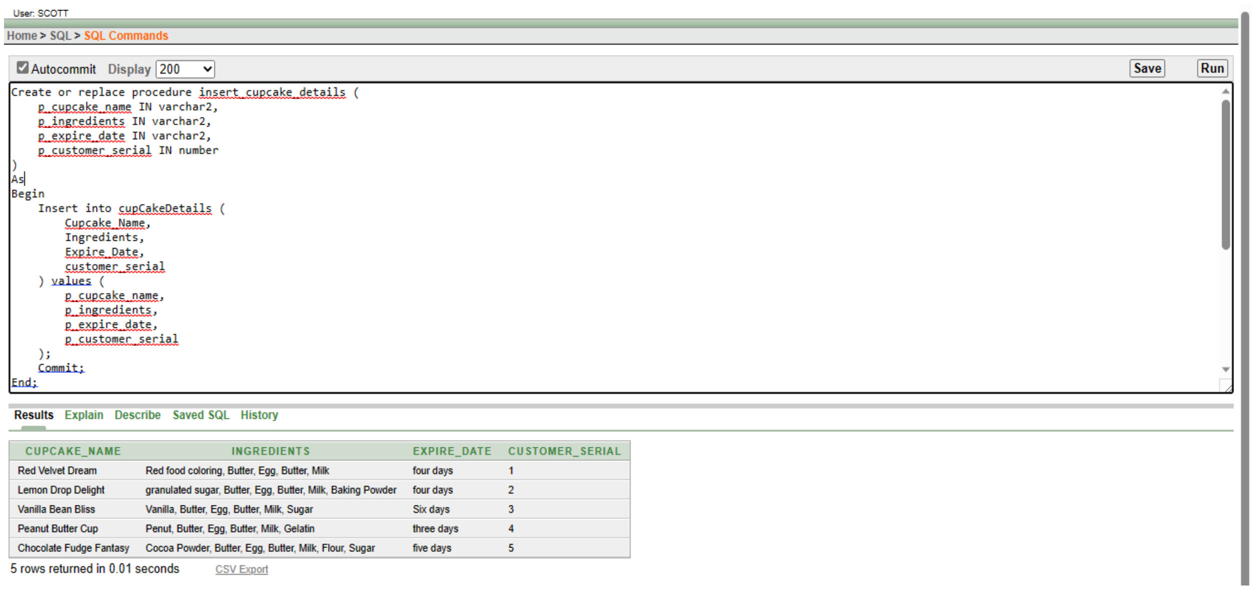
```
Total Bill Amount: 665

Statement processed.
```

0.02 seconds

Procedure

4. **Create a Procedure to insert a new row into the cupCakeDetails table, with a value for each column in the table.**

```
Create or replace procedure insert_cupcake_details ( p_cupcake_name IN varchar2, p_ingredients IN varchar2,  p_expire_date IN
varchar2, p_customer_serial IN number
)
as
Begin
   Insert into cupCakeDetails ( Cupcake_Name, Ingredients, Expire_Date,customer_serial)
    values ( p_cupcake_name,p_ingredients,  p_expire_date, p_customer_serial);
   Commit;
End;

Begin
   insert_cupcake_details(
      'Chocolate Fudge Fantasy',
      'Cocoa Powder, Butter, Egg, Butter, Milk, Flour, Sugar',
      'five days',
      5
   );
End;
```


User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 200
Save
Run
Results Explain Describe Saved SQL History
CSV Export

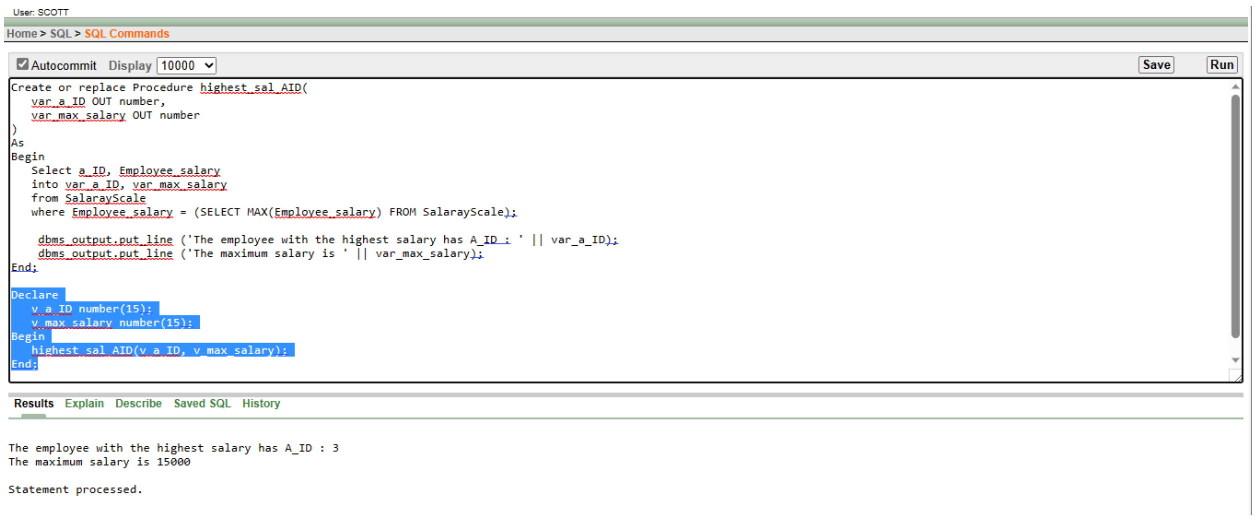
```
Create or replace procedure insert_cupcake_details (
    p_cupcake_name IN varchar2,
    p_ingredients IN varchar2,
    p_expire_date IN varchar2,
    p_customer_serial IN number
)
As
Begin
    Insert into cupCakeDetails (
        Cupcake_Name,
        Ingredients,
        Expire_Date,
        customer_serial
    ) values (
        p_cupcake_name,
        p_ingredients,
        p_expire_date,
        p_customer_serial
    );
    Commit;
End;
```

CUPCAKE_NAME	INGREDIENTS	EXPIRE_DATE	CUSTOMER_SERIAL
Red Velvet Dream	Red food coloring, Butter, Egg, Butter, Milk	four days	1
Lemon Drop Delight	granulated sugar, Butter, Egg, Butter, Milk, Baking Powder	four days	2
Vanilla Bean Bliss	Vanilla, Butter, Egg, Butter, Milk, Sugar	Six days	3
Peanut Butter Cup	Penut, Butter, Egg, Butter, Milk, Gelatin	three days	4
Chocolate Fudge Fantasy	Cocoa Powder, Butter, Egg, Butter, Milk, Flour, Sugar	five days	5

5 rows returned in 0.01 seconds

5. **Create a procedure to find A_ID of an employee with maximum amount of salary from Salarayscale Table.**

```
Create or replace Procedure highest_sal_AID(
  var_a_ID OUT number,
  var_max_salary OUT number
)
As
Begin
  Select a_ID, Employee_salary
  into var_a_ID, var_max_salary
  from SalarayScale
  where Employee_salary = (SELECT MAX(Employee_salary) FROM SalarayScale);

   dbms_output.put_line ('The employee with the highest salary has A_ID : ' || var_a_ID);
   dbms_output.put_line ('The maximum salary is ' || var_max_salary);
End;

Declare
  v_a_ID number(15);
  v_max_salary number(15);
Begin
  highest_sal_AID(v_a_ID, v_max_salary);
End;
```

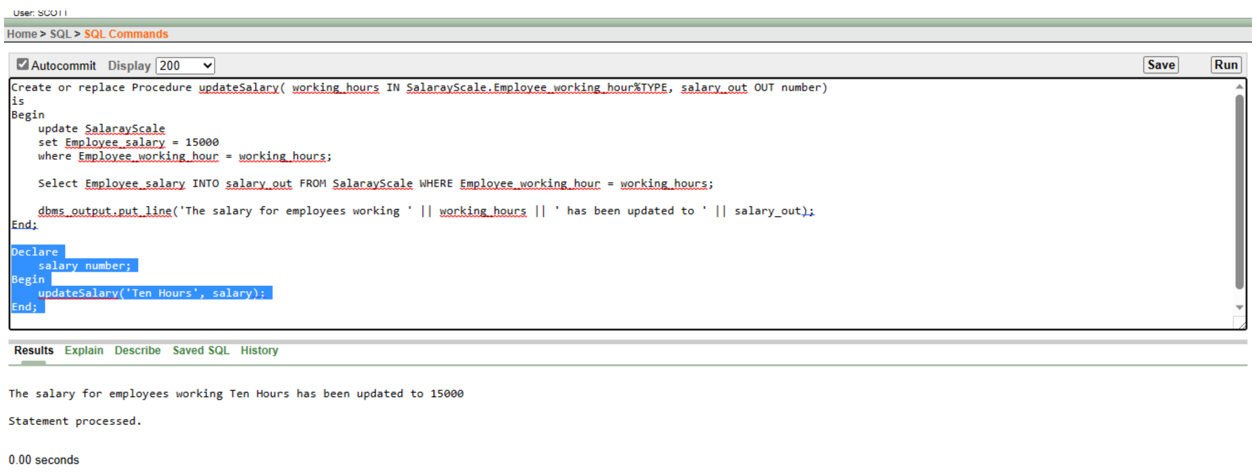

User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 10000
Save
Run
Results Explain Describe Saved SQL History

```
Create or replace Procedure highest_sal_AID(
   var_a_ID OUT number,
   var_max_salary OUT number
)
As
Begin
   Select a_ID, Employee_salary
   into var_a_ID, var_max_salary
   from SalarayScale
   where Employee_salary = (SELECT MAX(Employee_salary) FROM SalarayScale);

    dbms_output.put_line ('The employee with the highest salary has A_ID : ' || var_a_ID);
    dbms_output.put_line ('The maximum salary is ' || var_max_salary);
End;

Declare
   v_a_ID number(15);
   v_max_salary number(15);
Begin
   highest_sal_AID(v_a_ID, v_max_salary);
End;
```

```
The employee with the highest salary has A_ID : 3
The maximum salary is 15000

Statement processed.
```

6. Create a procedure that update a employees salary to 15000 from SalarayScale table who work 'Ten Hours' per day.

```
Create or replace Procedure updateSalary(
                    working_hours IN SalarayScale.Employee_working_hour%TYPE,
                     salary_out OUT number)
is
Begin
   update SalarayScale
   set Employee_salary = 15000
   where Employee_working_hour = working_hours;

   Select Employee_salary INTO salary_out FROM SalarayScale WHERE Employee_working_hour = working_hours;

   dbms_output.put_line('The salary for employees working ' || working_hours || ' has been updated to ' || salary_out);
End;

Declare
   salary number;
Begin
   updateSalary('Ten Hours', salary);
End;
```



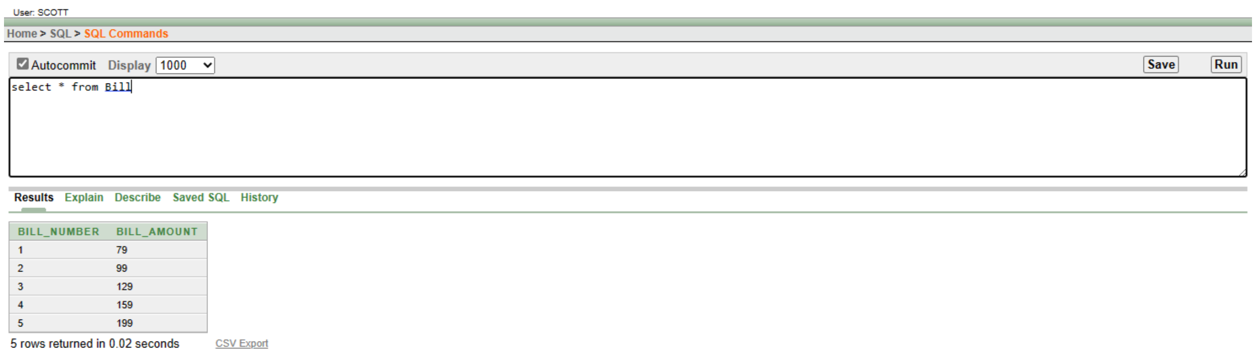
```
Create or replace Procedure updateSalary( working_hours IN SalarayScale.Employee_working_hour%TYPE, salary_out OUT number)
is
Begin
    update SalarayScale
    set Employee_salary = 15000
    where Employee_working_hour = working_hours;

    Select Employee_salary INTO salary_out FROM SalarayScale WHERE Employee_working_hour = working_hours;

    dbms_output.put_line('The salary for employees working ' || working_hours || ' has been updated to ' || salary_out);
End;

Declare
    salary number;
Begin
    updateSalary('Ten Hours', salary);
End;
```

```
The salary for employees working Ten Hours has been updated to 15000

Statement processed.

0.00 seconds
```

Trigger

1. **Create a trigger that updates the bill by adding VAT after a payment is done in the Bill table.**

```
Create or replace Trigger update_bill_with_vat
Before Update OF Bill_Amount ON Bill
for each row
Declare
 vat_amount Number(5) := 0.2;
Begin
 if :NEW.Bill_Amount > :OLD.Bill_Amount THEN
   :NEW.Bill_Amount := :NEW.Bill_Amount + (:NEW.Bill_Amount * vat_amount);
 End if;
End;

Update Bill
set Bill_Amount = 120
where Bill_Number = 1;
```

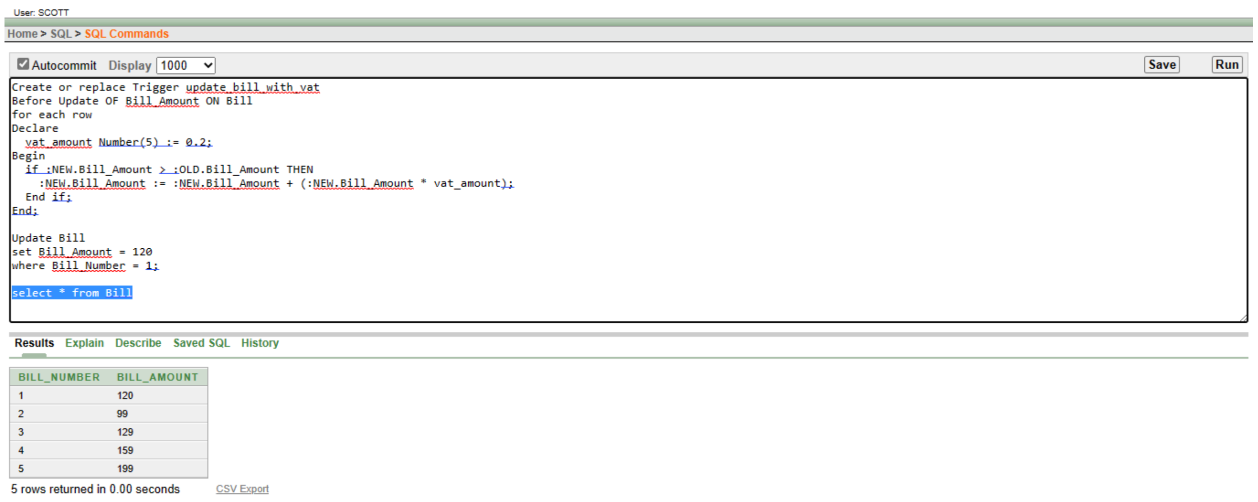

User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 1000
Save
Run

```
select * from Bill
```

Results Explain Describe Saved SQL History

BILL_NUMBER	BILL_AMOUNT
1	79
2	99
3	129
4	159
5	199

5 rows returned in 0.02 seconds   CSV Export


User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 1000
Save
Run
Results Explain Describe Saved SQL History
CSV Export

```
Create or replace Trigger update_bill_with_vat
Before Update OF Bill_Amount ON Bill
for each row
Declare
  vat_amount Number(5) := 0.2;
Begin
  if :NEW.Bill_Amount > :OLD.Bill_Amount THEN
    :NEW.Bill_Amount := :NEW.Bill_Amount + (:NEW.Bill_Amount * vat_amount);
  End if;
End;

Update Bill
set Bill_Amount = 120
where Bill_Number = 1;

select * from Bill
```

BILL_NUMBER	BILL_AMOUNT
1	120
2	99
3	129
4	159
5	199

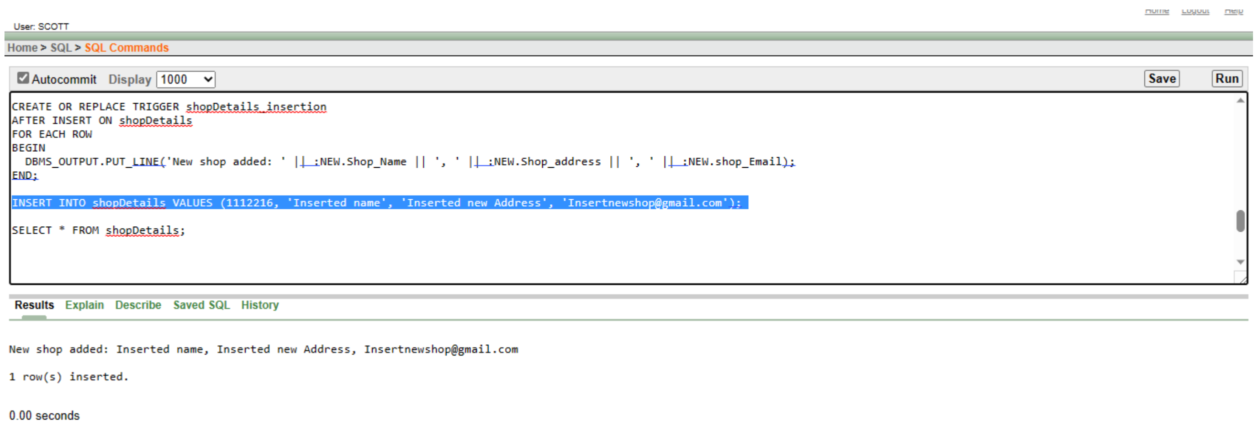
5 rows returned in 0.00 seconds

2. Create a trigger that Insert a new row in each column in shopDetails table.

CREATE OR REPLACE TRIGGER shopDetails_insertion
AFTER INSERT ON shopDetails
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('New shop added: ' || :NEW.Shop_Name || ', ' || :NEW.Shop_address || ', ' || :NEW.shop_Email);
END;

INSERT INTO shopDetails VALUES (1112216, 'Inserted name', 'Inserted new Address', 'Insertnewshop@gmail.com');

SELECT * FROM shopDetails;

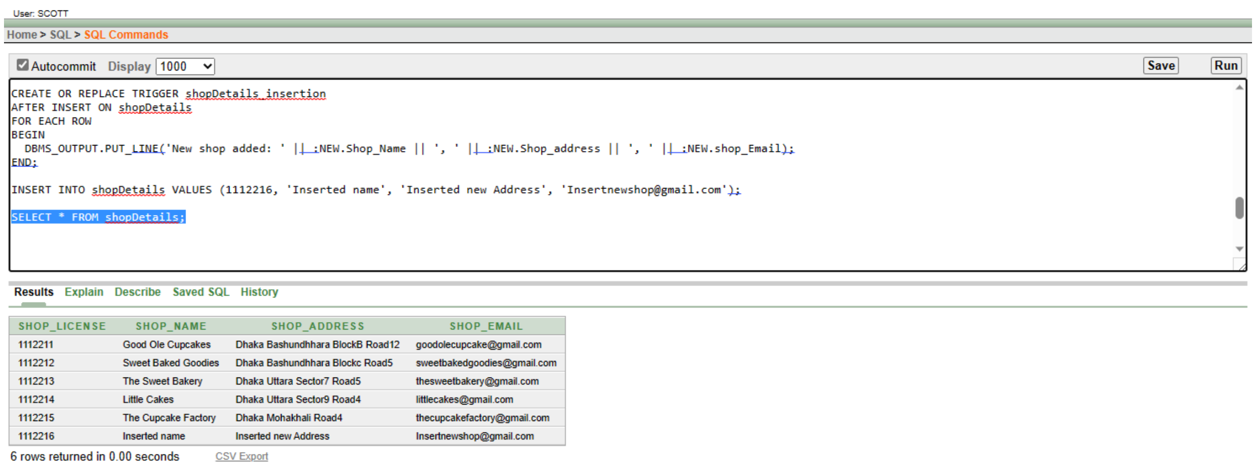

Home Logout Help
User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 1000
Save
Run
Results Explain Describe Saved SQL History

```
CREATE OR REPLACE TRIGGER shopDetails_insertion
AFTER INSERT ON shopDetails
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('New shop added: ' || :NEW.Shop_Name || ', ' || :NEW.Shop_address || ', ' || :NEW.shop_Email);
END;

INSERT INTO shopDetails VALUES (1112216, 'Inserted name', 'Inserted new Address', 'Insertnewshop@gmail.com');

SELECT * FROM shopDetails;
```

```
New shop added: Inserted name, Inserted new Address, Insertnewshop@gmail.com

1 row(s) inserted.
```

0.00 seconds

<u>Package</u>


User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 1000
Save
Run

```
CREATE OR REPLACE TRIGGER shopDetails_insertion
AFTER INSERT ON shopDetails
FOR EACH ROW
BEGIN
  DBMS_OUTPUT.PUT_LINE('New shop added: ' || :NEW.Shop_Name || ', ' || :NEW.Shop_address || ', ' || :NEW.shop_Email);
END;

INSERT INTO shopDetails VALUES (1112216, 'Inserted name', 'Inserted new Address', 'Insertnewshop@gmail.com');

SELECT * FROM shopDetails;
```

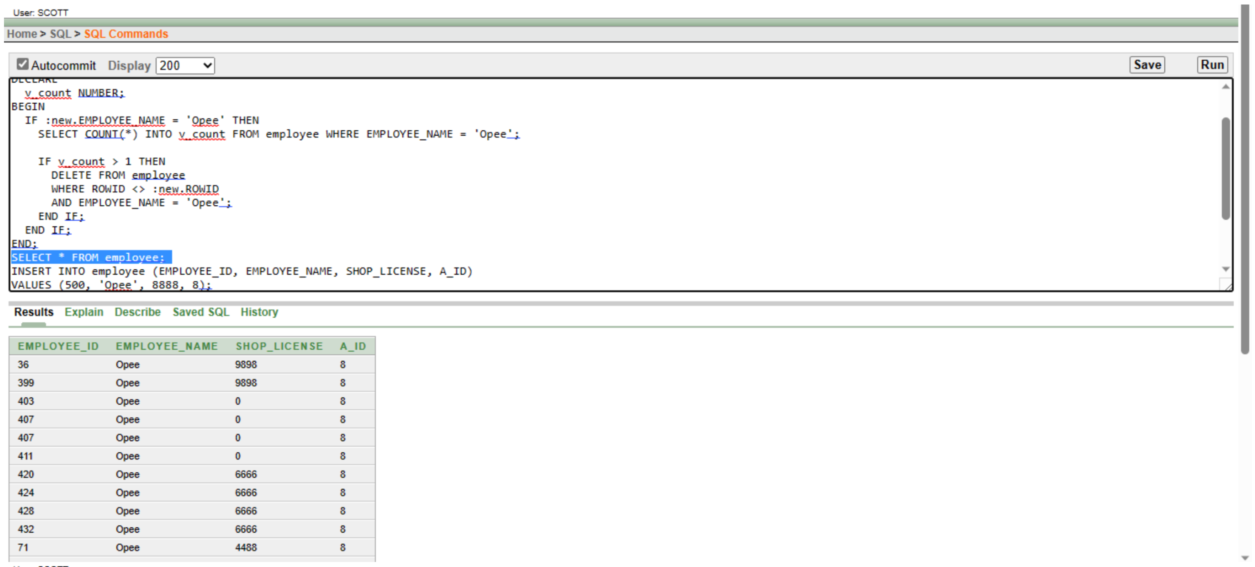
Results Explain Describe Saved SQL History

SHOP_LICENSE	SHOP_NAME	SHOP_ADDRESS	SHOP_EMAIL
1112211	Good Ole Cupcakes	Dhaka Bashundhhara BlockB Road12	goodolecupcake@gmail.com
1112212	Sweet Baked Goodies	Dhaka Bashundhhara Blockc Road5	sweetbakedgoodies@gmail.com
1112213	The Sweet Bakery	Dhaka Uttara Sector7 Road5	thesweetbakery@gmail.com
1112214	Little Cakes	Dhaka Uttara Sector9 Road4	littlecakes@gmail.com
1112215	The Cupcake Factory	Dhaka Mohakhali Road4	thecupcakefactory@gmail.com
1112216	Inserted name	Inserted new Address	Insertnewshop@gmail.com

6 rows returned in 0.00 seconds CSV Export

3. Create a trigger that will delete any duplicate rows with EMPLOYEE_NAME = 'Opee' leaving only one row with that value.

```
CREATE OR REPLACE TRIGGER delete_duplicate_opee
BEFORE INSERT OR UPDATE ON employee
FOR EACH ROW
DECLARE
  v_count NUMBER;
BEGIN
  IF :new.EMPLOYEE_NAME = 'Opee' THEN
    SELECT COUNT(*) INTO v_count FROM employee WHERE EMPLOYEE_NAME = 'Opee';

    IF v_count > 1 THEN
      DELETE FROM employee
      WHERE ROWID <> :new.ROWID
      AND EMPLOYEE_NAME = 'Opee';
    END IF;
  END IF;
END;
ALTER TRIGGER delete_duplicate_opee ENABLE;
SELECT * FROM employee;
INSERT INTO employee (EMPLOYEE_ID, EMPLOYEE_NAME, SHOP_LICENSE, A_ID)
VALUES (500, 'Opee', 8888, 8);

SELECT * FROM employee;
```



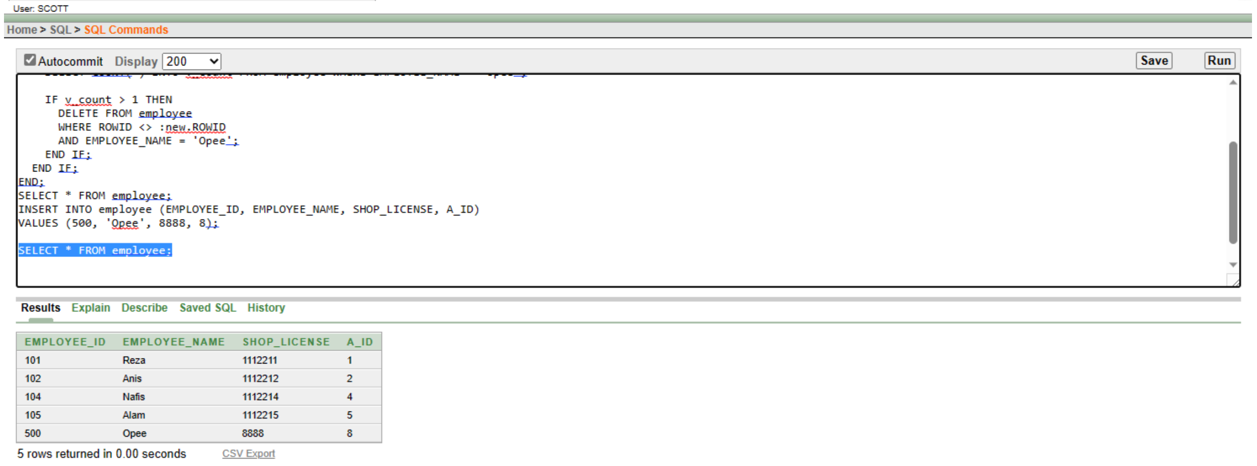
User: SCOTT

Home > SQL > SQL Commands

Autocommit Display 200

```
DECLARE
  v_count NUMBER;
BEGIN
  IF :new.EMPLOYEE_NAME = 'Opee' THEN
    SELECT COUNT(*) INTO v_count FROM employee WHERE EMPLOYEE_NAME = 'Opee';

    IF v_count > 1 THEN
      DELETE FROM employee
      WHERE ROWID <> :new.ROWID
      AND EMPLOYEE_NAME = 'Opee';
    END IF;
  END IF;
END;
SELECT * FROM employee;
INSERT INTO employee (EMPLOYEE_ID, EMPLOYEE_NAME, SHOP_LICENSE, A_ID)
VALUES (500, 'Opee', 8888, 8);
```

EMPLOYEE_ID	EMPLOYEE_NAME	SHOP_LICENSE	A_ID
36	Opee	9898	8
399	Opee	9898	8
403	Opee	0	8
407	Opee	0	8
407	Opee	0	8
411	Opee	0	8
420	Opee	6666	8
424	Opee	6666	8
428	Opee	6666	8
432	Opee	6666	8
71	Opee	4488	8



User: SCOTT

Home > SQL > SQL Commands

Autocommit Display 200

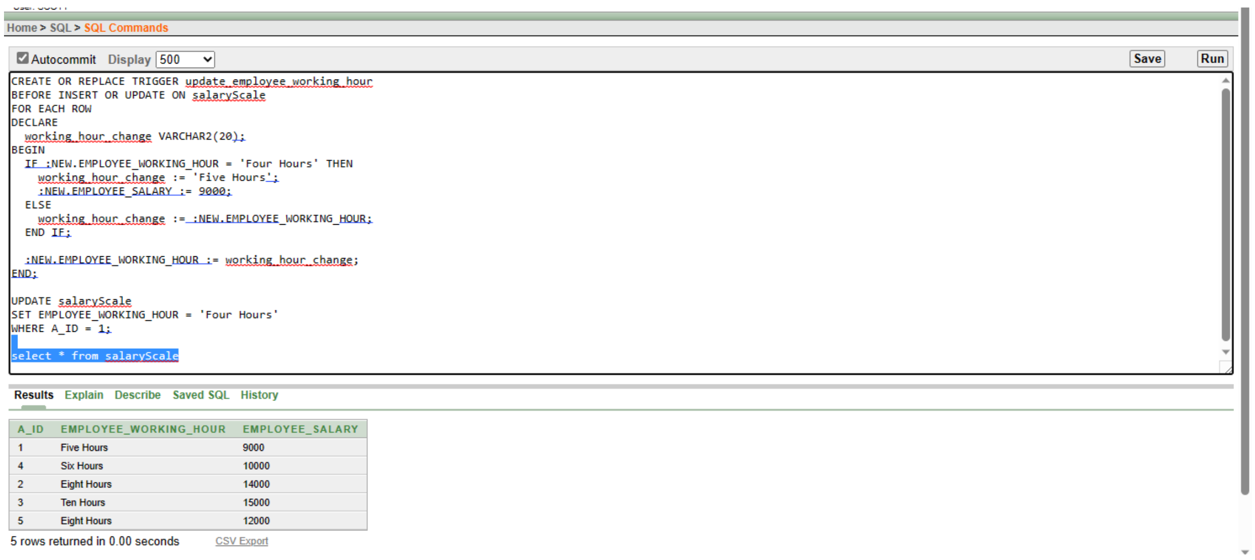
```

    IF v_count > 1 THEN
      DELETE FROM employee
      WHERE ROWID <> :new.ROWID
      AND EMPLOYEE_NAME = 'Opee';
    END IF;
  END IF;
END;
SELECT * FROM employee;
INSERT INTO employee (EMPLOYEE_ID, EMPLOYEE_NAME, SHOP_LICENSE, A_ID)
VALUES (500, 'Opee', 8888, 8);

SELECT * FROM employee;
```

EMPLOYEE_ID	EMPLOYEE_NAME	SHOP_LICENSE	A_ID
101	Reza	1112211	1
102	Anis	1112212	2
104	Nafis	1112214	4
105	Alam	1112215	5
500	Opee	8888	8

5 rows returned in 0.00 seconds   CSV Export

4. Create a trigger that will delete any duplicate rows with EMPLOYEE_NAME = 'Opee' leaving only one row with that value.

```
CREATE OR REPLACE TRIGGER update_employee_working_hour
BEFORE INSERT OR UPDATE ON salarayScale
FOR EACH ROW
DECLARE
 working_hour_change VARCHAR2(20);
BEGIN
 IF :NEW.EMPLOYEE_WORKING_HOUR = 'Four Hours' THEN
  working_hour_change := 'Five Hours';
  :NEW.EMPLOYEE_SALARY := 9000;
 ELSE
  working_hour_change := :NEW.EMPLOYEE_WORKING_HOUR;
 END IF;
 :NEW.EMPLOYEE_WORKING_HOUR := working_hour_change;
END;
UPDATE salarayScale
SET EMPLOYEE_WORKING_HOUR = 'Four Hours'
WHERE A_ID = 1;

select * from salarayScale
```


Home > SQL > SQL Commands

Autocommit Display 500 Save Run

```
CREATE OR REPLACE TRIGGER update_employee_working_hour
BEFORE INSERT OR UPDATE ON salaryScale
FOR EACH ROW
DECLARE
  working_hour_change VARCHAR2(20);
BEGIN
  IF :NEW.EMPLOYEE_WORKING_HOUR = 'Four Hours' THEN
    working_hour_change := 'Five Hours';
    :NEW.EMPLOYEE_SALARY := 9000;
  ELSE
    working_hour_change := :NEW.EMPLOYEE_WORKING_HOUR;
  END IF;

  :NEW.EMPLOYEE_WORKING_HOUR := working_hour_change;
END;

UPDATE salaryScale
SET EMPLOYEE_WORKING_HOUR = 'Four Hours'
WHERE A_ID = 1;

select * from salaryScale
```

Results Explain Describe Saved SQL History

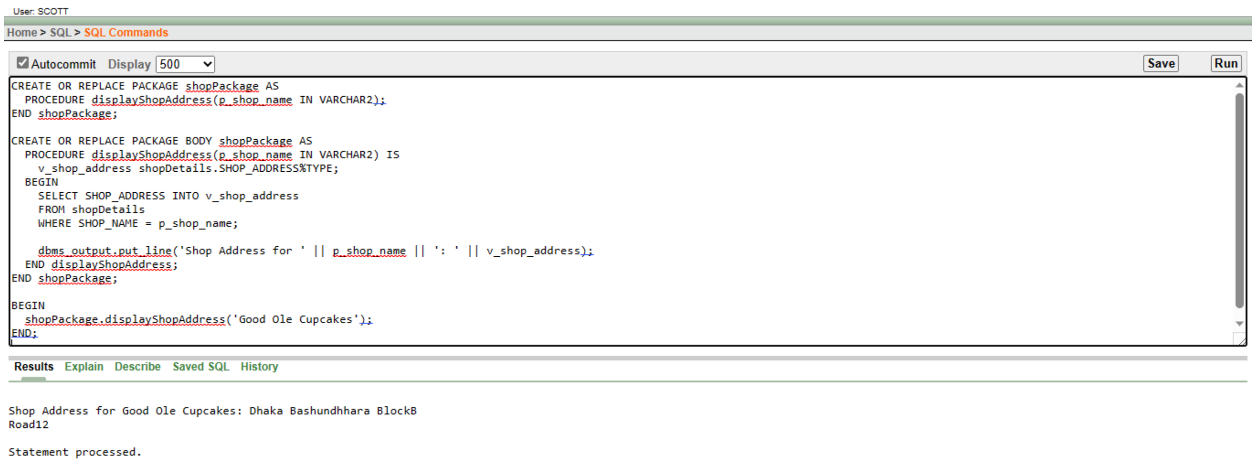
A_ID	EMPLOYEE_WORKING_HOUR	EMPLOYEE_SALARY
1	Five Hours	9000
4	Six Hours	10000
2	Eight Hours	14000
3	Ten Hours	15000
5	Eight Hours	12000

5 rows returned in 0.00 seconds CSV Export

Package

1. **Creating a package with a procedure that displays the SHOP_ADDRESS based on the SHOP_NAME passed as a parameter**

```
CREATE OR REPLACE PACKAGE shopPackage AS
  PROCEDURE displayShopAddress(p_shop_name IN VARCHAR2);
END shopPackage;

CREATE OR REPLACE PACKAGE BODY shopPackage AS
  PROCEDURE displayShopAddress(p_shop_name IN VARCHAR2) IS
    v_shop_address shopDetails.SHOP_ADDRESS%TYPE;
  BEGIN
    SELECT SHOP_ADDRESS INTO v_shop_address
    FROM shopDetails
    WHERE SHOP_NAME = p_shop_name;

    dbms_output.put_line('Shop Address for ' || p_shop_name || ': ' || v_shop_address);
  END displayShopAddress;
END shopPackage;

BEGIN
  shopPackage.displayShopAddress('Good Ole Cupcakes');
END;
```


User: SCOTT

Home > SQL > SQL Commands

Autocommit Display 500 Save Run

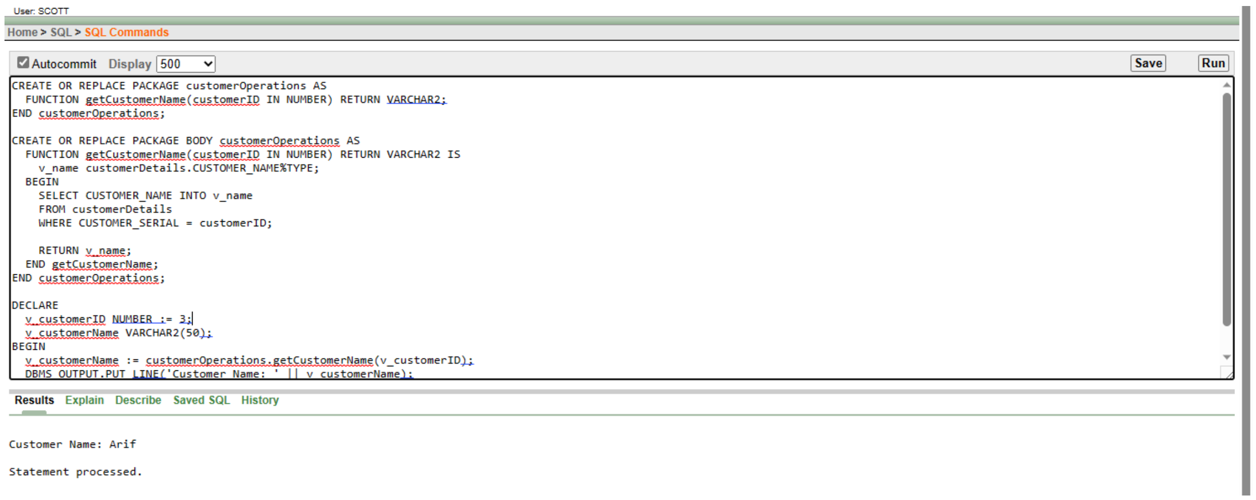
```
CREATE OR REPLACE PACKAGE shopPackage AS
  PROCEDURE displayShopAddress(p_shop_name IN VARCHAR2);
END shopPackage;

CREATE OR REPLACE PACKAGE BODY shopPackage AS
  PROCEDURE displayShopAddress(p_shop_name IN VARCHAR2) IS
    v_shop_address shopDetails.SHOP_ADDRESS%TYPE;
  BEGIN
    SELECT SHOP_ADDRESS INTO v_shop_address
    FROM shopDetails
    WHERE SHOP_NAME = p_shop_name;

    dbms_output.put_line('Shop Address for ' || p_shop_name || ': ' || v_shop_address);
  END displayShopAddress;
END shopPackage;

BEGIN
  shopPackage.displayShopAddress('Good Ole Cupcakes');
END;
```

Results Explain Describe Saved SQL History

```
Shop Address for Good Ole Cupcakes: Dhaka Bashundhhara BlockB
Road12

Statement processed.
```

2. Create a package with a function that displays the Customer_name based on the Customer_ID passed as a parameter?

```
CREATE OR REPLACE PACKAGE customerOperations AS

 FUNCTION getCustomerName(customerID IN NUMBER) RETURN VARCHAR2;

END customerOperations;

CREATE OR REPLACE PACKAGE BODY customerOperations AS

 FUNCTION getCustomerName(customerID IN NUMBER) RETURN VARCHAR2 IS

  v_name customerDetails.CUSTOMER_NAME%TYPE;

 BEGIN

  SELECT CUSTOMER_NAME INTO v_name

  FROM customerDetails

  WHERE CUSTOMER_SERIAL = customerID;

  RETURN v_name;

 END getCustomerName;

END customerOperations;

DECLARE

 v_customerID NUMBER := 3;

 v_customerName VARCHAR2(50);

BEGIN

 v_customerName := customerOperations.getCustomerName(v_customerID);

 DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customerName);

END;
```


User: SCOTT
Home > SQL > SQL Commands
Autocommit Display 500
Save
Run
Results Explain Describe Saved SQL History

```
CREATE OR REPLACE PACKAGE customerOperations AS
  FUNCTION getCustomerName(customerID IN NUMBER) RETURN VARCHAR2;
END customerOperations;

CREATE OR REPLACE PACKAGE BODY customerOperations AS
  FUNCTION getCustomerName(customerID IN NUMBER) RETURN VARCHAR2 IS
    v_name customerDetails.CUSTOMER_NAME%TYPE;
  BEGIN
    SELECT CUSTOMER_NAME INTO v_name
    FROM customerDetails
    WHERE CUSTOMER_SERIAL = customerID;

    RETURN v_name;
  END getCustomerName;
END customerOperations;

DECLARE
  v_customerID NUMBER := 3;
  v_customerName VARCHAR2(50);
BEGIN
  v_customerName := customerOperations.getCustomerName(v_customerID);
  DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customerName);
```

```
Customer Name: Arif

Statement processed.
```