



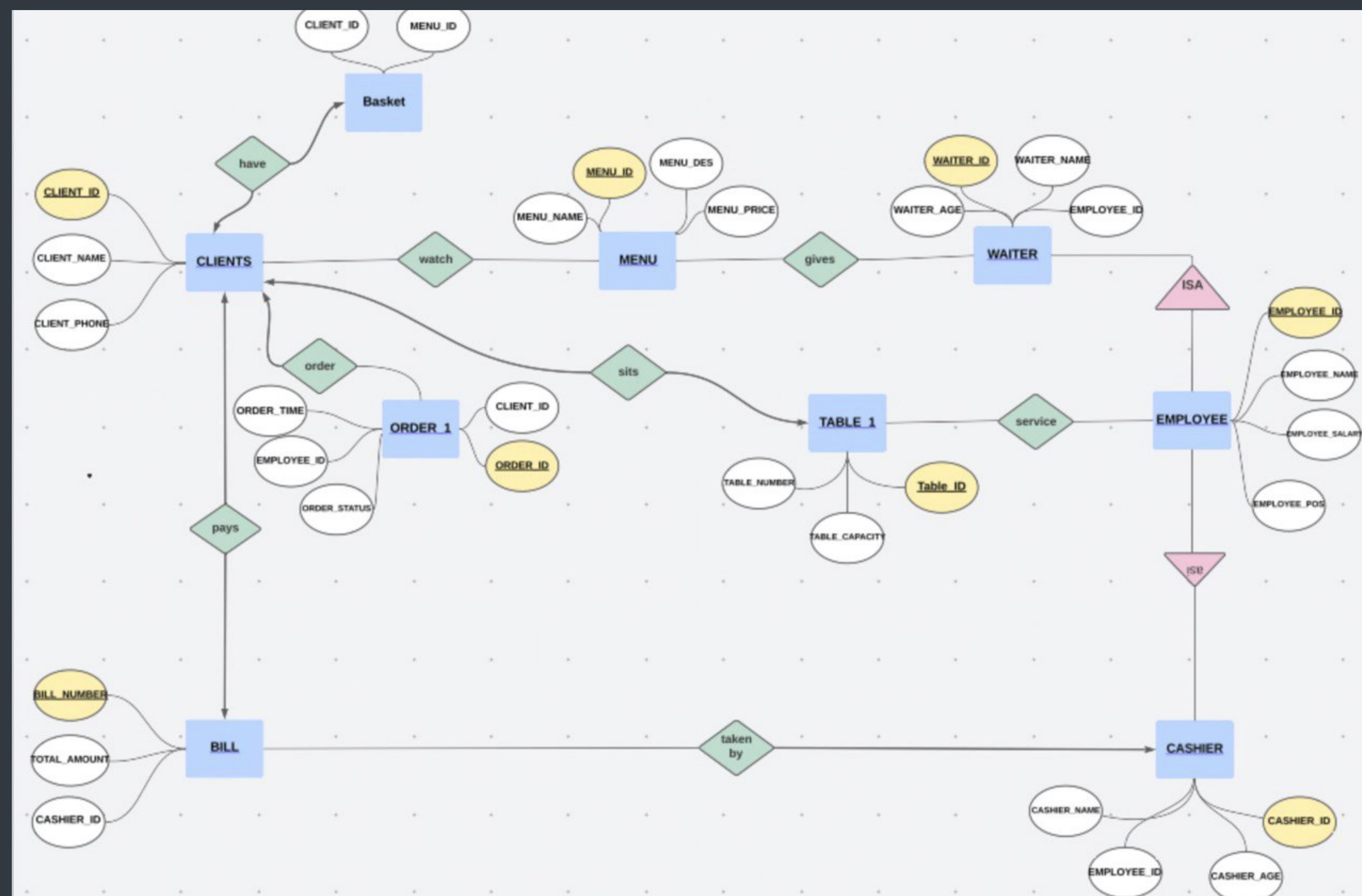
---

# COFFEE SHOP DATABASE MANAGEMENT

This system provides client service, order, and employee performance at the coffee shop. Also helps to find a quick access to any information such as clients, employees, menu, etc. It shows the step from the client's visit to the client's payment of the bill.

Ingkar Koilybai, Nurai Abugalieva,  
Aknur Satybay, Nuray Zhumabay, Ali Maksotov

# ERD



## Table Client

{Client\_id}=>{Client\_name},{Client\_phone}

Client\_id->p key

CLIENT_ID	CLIENT_NAME	CLIENT_PHONE
2	Remy	248-647-8968
4	Debby	412-139-2609
5	Kelvy	304-774-0196

EMPLOYEE_ID	EMPLOYEE_NAME	EMPLOYEE_POS	EMPLOYEE_SALARY
1	Rogers	chef	110000
25	Glynn	chef	110000
26	Damita	chef	110000
15	Aisha	waiter	110000

## Table Employee

{Employee\_id}->{Employee\_name},{Employee\_pos},

{Employee\_salary}

Employee\_id->p key

## Table Waiter(subclass of Employee)

{Waiter\_id},{Emp\_id}->{Waiter\_name},  
 {Waiter\_age}

Waiter\_id->p key

Emp\_id->foreign key

WAITER_ID	WAITER_NAME	WAITER_AGE	EMPLOYEE_ID
20	Aisha	30	15
21	Aisha	18	16
22	Akhan	22	17
23	Alisher	19	18

Copyright © 1999, 2023, Oracle and/or its affiliates.

CASHIER_ID	CASHIER_NAME	CASHIER_AGE	EMPLOYEE_ID
200	Nana	18	30
2	Boysey	24	2
3	Karel	31	3
202	Sabrina	20	32

## Table Cashier(subclass of Employee)

{Cashier\_id},{Emp\_id}->{Cashier\_name},{Cashier\_age}

Cashier\_id->p key

Emp\_id->foreign key

## Table Bill

- Bill\_number, Cashier\_id, Client\_id -> Total\_amount
- Bill\_number, Cashier\_id, Client\_id -> Payment\_method

BILL_NUMBER	TOTAL_AMOUNT	CASHIER_ID	CLIENT_ID	PAYMENT_METHOD
70	1300	200	1	CARD
71	11000	201	2	CASH
72	20000	202	3	CARD

MENU_ID	MENU_DESCRIPTION	MENU_NAME	MENU_PRICE
2	boloneze	pasta	1790
3	ice limonade	moxito	1790
4	classic	black tea	690
5	with meat	pelmeni	1490

**Table Menu**

{Menu\_id}->{Menu\_name},  
{Menu\_description,Menu\_price}  
Menu\_id->p key

**Table Order\_1**

{Order\_id},{Emp\_id},{Client\_id}->  
{Order\_time},{Order\_status}  
Emp\_id,Client\_id->foreign keys

Results	Explain	Describe	Saved SQL	History	ORDER_ID	ORDER_STATUS	ORDER_TIME	CLIENT_ID	TABLE_ID	EMPLOYEE_ID
					51	reservation	0:16	99	60	8
					59	runout	12:00	7	68	1
					40	ready	12:00	1	60	1

**Table Table\_1**

{Table\_id}->{Table\_number},  
{Table\_capacity}  
Table\_id->p key

TABLE_ID	TABLE_NUMBER	TABLE_CAPACITY
60	1	15
61	2	10
62	3	15
63	4	20
64	5	5

**Table Basket**

Client\_id,Menu\_ID->foreign key

CLIENT_ID	MENU_ID
1	2
1	5
1	2

# FUNCTION

creating a function  
which calculates  
total amount of  
food that client  
ordered

The screenshot shows the Oracle SQL Developer interface with the following code in the SQL worksheet:

```
1 create or replace function f3 return VARCHAR2 is
2 TOTAL integer := 0;
3 t varchar(32767);
4 tt integer := 0;
5 total_clients integer;
6 i integer := 1;
7 b boolean := FALSE;
8 cursor c is
9 select CLIENT_ID, MENU_ID
10 from basket;
11 begin
12     select count(*) into total_clients
13     from clients;
14     while i <= total_clients loop
15         b := FALSE;
16         for row in c loop
17             if row.CLIENT_ID = i then
18                 b := TRUE;
19             end if;
20         end loop;
21         if b = TRUE then
22             for row in c loop
23                 if row.CLIENT_ID = i then
24                     select MENU_PRICE into tt from menu where menu_id = row.MENU_ID;
25                     TOTAL := TOTAL + tt;
26                 end if;
27             end loop;
28             t := t || CHR(10) || 'Client ID is ' || i || ' and his total is ' || TOTAL;
29             TOTAL := 0;
30             tt := 0;
31         end if;
32         i := i + 1;
33     end loop;
34     return t;
35 end;
36
37 DECLARE
38     n varchar2(32767);
39 BEGIN
40     n:=f3();
41     dbms_output.put_line(n);
42 END;
43
```

The results pane shows the output of the function execution:

Results	Explain	Describe	Saved SQL	History
Client ID is 1 and his total is 5070 Client ID is 3 and his total is 1790				

# FUNCTION 2

CREATING A FUNCTION  
WHICH SHOWS NUMBER  
OF CASHIERS WHO ARE  
UNDER 18

```
1 CREATE OR REPLACE FUNCTION cashierAge
2 RETURN NUMBER IS
3 counts number(20):=0;
4
5 begin
6     SELECT count(*) into counts FROM CASHIER WHERE CASHIER_AGE<18 ;
7     RETURN counts;
8
9 EXCEPTION
10 WHEN NO_DATA_FOUND
11 THEN
12     RETURN 0;
13 WHEN OTHERS
14 THEN
15     RETURN -1;
16 END;
17
18 DECLARE
19     n number(20);
20 BEGIN
21     n := cashierAge();
22     dbms_output.put_line('Total number of cashiers who are under 18 is ' || n);
23 END;
24
```

Results Explain Describe Saved SQL History

Total number of cashiers who are under 18 is 2

Statement processed.

# Trigger 1

Oracle APEX

SQL Workshop Team Development Gallery

Search

DM DATABASE2 MID... sdu

Schema WKSP\_SDU

TR1

Code Errors Object Details DDL

Download Save and Compile Drop Refresh

A..

```
1 create or replace trigger tr1
2 before insert on order_1
3 for each row
4 declare
5   order_statusk order_1.ORDER_STATUS%type := :new.order_status;
6   order_idk order_1.ORDER_ID%type := :new.order_id;
7   order_timek order_1.ORDER_TIME%type := :new.order_time;
8   hour_str varchar2(255);
9   minus_str varchar2(255);
10  hour_int integer;
11  minus_int integer;
12  total_order integer;
13  h_int integer;
14  m_int integer;
15
16  cursor c is
17    select *
18    from order_1
19    where order_status = 'reservation' and table_id = :new.table_id;
20 begin
21
22  DBMS_OUTPUT.PUT_LINE('Order_ID: ' || order_idk || ' | Order_Status: ' || order_statusk);
23  hour_str := SUBSTR(order_timek, 1, INSTR(order_timek, ':') - 1);
24  minus_str := SUBSTR(order_timek, INSTR(order_timek, ':') + 1);
25  DBMS_OUTPUT.PUT_LINE(hour_str || ' ' || minus_str);
26  hour_int := TO_NUMBER(hour_str);
27  minus_int := TO_NUMBER(minus_str);
28
29  select count(*) into total_order
30  from order_1
31  where order_status = 'reservation' and client_id = :new.client_id;
32
33  DBMS_OUTPUT.PUT_LINE(total_order);
34
35  if :new.order_status != 'runout' then
36    if total_order = 1 then
37      RAISE_APPLICATION_ERROR (-20001, 'Ol klientte uje reservnii stol bar');
38    else
39      for row in c loop
40        h_int := TO_NUMBER(SUBSTR(row.order_time, 1, INSTR(row.order_time, ':') - 1));
41        m_int := TO_NUMBER(SUBSTR(row.order_time, INSTR(row.order_time, ':') + 1));
42        if (h_int + 1) = hour_int or (h_int = hour_int) or (hour_int + 1) = h_int then
43          RAISE_APPLICATION_ERROR (-20001, 'Uje reservnii uikit bar ol stolikta');
44        elsif ((h_int + 2) = hour_int) and (m_int >= minus_int) or (hour_int + 2 = h_int and minus_int >= m_int) then
45          RAISE_APPLICATION_ERROR (-20001, 'Uje reservnii uikit bar ol stolikta');
46      end if;
47    end loop;
48  end if;
49  end if;
50
```

Copyright © 1999, 2023, Oracle and/or its affiliates.

ПОИСК

SDU Gmail Leetcode Oracle APEX Books

APEX App Builder SQL Workshop Team Development Gallery

Search

DM DATABASE2 MID... sdu

Schema WKSP\_SDU

Object Browser

Type to filter...

TR1

Views Indexes Sequences Types Packages Procedures Functions Triggers

Code Errors Object Details DDL

Download Save and Compile Drop Refresh

A..

```
22  DBMS_OUTPUT.PUT_LINE('Order_ID: ' || order_idk || ' | Order_Status: ' || order_statusk);
23  hour_str := SUBSTR(order_timek, 1, INSTR(order_timek, ':') - 1);
24  minus_str := SUBSTR(order_timek, INSTR(order_timek, ':') + 1);
25  DBMS_OUTPUT.PUT_LINE(hour_str || ' ' || minus_str);
26  hour_int := TO_NUMBER(hour_str);
27  minus_int := TO_NUMBER(minus_str);
28
29  select count(*) into total_order
30  from order_1
31  where order_status = 'reservation' and client_id = :new.client_id;
32
33  DBMS_OUTPUT.PUT_LINE(total_order);
34
35  if :new.order_status != 'runout' then
36    if total_order = 1 then
37      RAISE_APPLICATION_ERROR (-20001, 'Ol klientte uje reservnii stol bar');
38    else
39      for row in c loop
40        h_int := TO_NUMBER(SUBSTR(row.order_time, 1, INSTR(row.order_time, ':') - 1));
41        m_int := TO_NUMBER(SUBSTR(row.order_time, INSTR(row.order_time, ':') + 1));
42        if (h_int + 1) = hour_int or (h_int = hour_int) or (hour_int + 1) = h_int then
43          RAISE_APPLICATION_ERROR (-20001, 'Uje reservnii uikit bar ol stolikta');
44        elsif ((h_int + 2) = hour_int) and (m_int >= minus_int) or (hour_int + 2 = h_int and minus_int >= m_int) then
45          RAISE_APPLICATION_ERROR (-20001, 'Uje reservnii uikit bar ol stolikta');
46      end if;
47    end loop;
48  end if;
49  end if;
50
```

Copyright © 1999, 2023, Oracle and/or its affiliates.

koilibai04@bk.ru sdu en

24°C Cloudy

ПОИСК

SDU Gmail Leetcode Oracle APEX Books

ENG 17:42 24.04.2023

# Trigger 2

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'Oracle APEX' and 'Books'. Below it are tabs for 'SQL Workshop' (selected), 'Team Development', and 'Gallery'. A search bar and a user icon are also present. The main area displays a trigger named 'TR2'. The 'Code' tab is selected, showing the following PL/SQL code:

```
1  create or replace trigger tr2
2  before insert on order_1
3  for each row
4  declare
5  idd number;
6  begin
7  if :new.order_status = 'runout' then
8  select order_id into idd
9  from order_1
10 where table_id = :new.table_id and :new.client_id = client_id and order_time = :new.order_time and order_status = 'reservation';
11 DELETE FROM order_1 WHERE order_id = idd;
12 DELETE FROM order_1 WHERE order_id = :new.order_id;
13 end if;
14 end;
15 /
```

The code editor includes standard toolbar buttons for 'Download', 'Save and Compile' (which is highlighted in green), 'Drop', and 'Refresh'. There are also icons for undo, redo, search, and schema selection.

# Trigger 3

SQL Workshop

Team Development

Gallery

Search



TR3

Code

Errors

Object Details

DDL

Download

Save and Compile

Drop

Ref



```
1  create or replace trigger tr3
2  before insert on order_1
3  for each row
4  declare
5  total_bill integer;
6  tt integer := 0;
7
8  function total_amountko(c_id number) return number is
9  cursor c_c is
10 select *
11 from basket;
```



TR3

Code

Errors

Object Details

DDL

Download

Save and Compile

Drop

Refresh



```
12 dish_price integer := 0;
13 total integer := 0;
14 begin
15 for row in c_c loop
16 if row.CLIENT_ID = c_id then
17   select menu_price into dish_price
18   from menu
19   where menu_id = row.menu_id;
20   total := total + dish_price;
21   dish_price := 0;
22 end if;
23 end loop;
24 return total;
25 end;
26
27 begin
28 if :new.order_status = 'runout' then
29   select count(*) into total_bill
30   from bill;
31   tt := total_amountko(:new.client_id);
32   insert into bill values(total_bill, tt, 200, :new.client_id, 'CARD');
33   delete from basket where client_id = :new.client_id;
34 end if;
35 end;
36
```

# 1 procedure

Language SQL ? Rows 10 ? Clear Co

C | Q | A: |

```
CREATE OR REPLACE PROCEDURE update_employee_salary(
    n_employee_id IN employee.employee_id%TYPE,
    i_perc IN NUMBER
)
IS
no_found exception;
BEGIN
    UPDATE employee
    SET employee_salary = employee_salary * (1 + i_perc/100)
    WHERE employee_id = n_employee_id;

    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'No employee found with ID ' || n_employee_id);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Salary updated for employee ' || n_employee_id || '.');
    END IF;
EXCEPTION
    WHEN no_found THEN
        dbms_output.put_line('No such employee!');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
END;
```

Results Explain Describe Saved SQL History

procedure created.

2

C | Q | A: |

```
1 BEGIN
2 | update_employee_salary(5, 10);
3 END;
4
5
```

Results Explain Describe Saved SQL History

Salary updated for employee 5.

Statement processed.

# procedura 2

```
1  CREATE OR REPLACE PROCEDURE find_sname
2    (o_employee_id IN NUMBER,
3     o_employee_name OUT VARCHAR2,
4     o_employee_pos OUT varchar2
5    )
6    AS
7    BEGIN
8      SELECT employee_name, employee_pos
9        INTO o_employee_name, o_employee_pos
10     FROM employee
11    WHERE employee_id = o_employee_id;
12    EXCEPTION
13    WHEN OTHERS
14    THEN
15      DBMS_OUTPUT.PUT_LINE('Error in finding employee_id: '||o_employee_id);
16    END find_sname;
17  |
18
```

[Results](#)   [Explain](#)   [Describe](#)   [Saved SQL](#)   [History](#)

Salary updated for employee 5.

Statement processed.

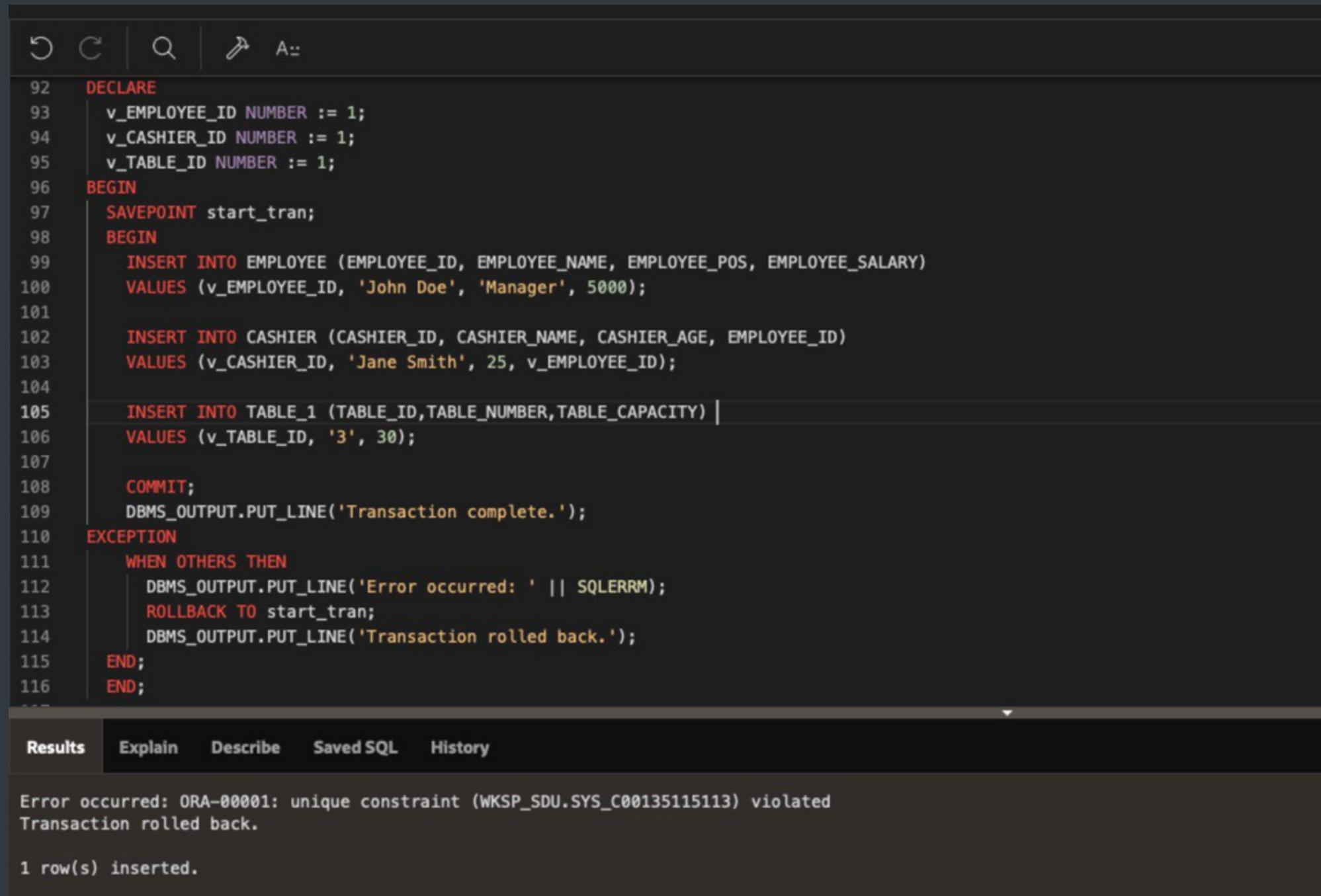
```
1 DECLARE
2   | v_local_employee_name employee.employee_name%TYPE;
3   | v_local_employee_pos employee.employee_pos%TYPE;
4 BEGIN
5   | find_employee(1, v_local_employee_name, v_local_employee_pos);
6   | DBMS_OUTPUT.PUT_LINE('Employee is: ' || v_local_employee_name || ' ' || v_local_employee_pos || '.');
7 END;
8 /
```

Results   Explain   Describe   Saved SQL   History

Employee is: Rogers chef.

Statement processed.

# Transaction 1



The screenshot shows a SQL developer interface with a dark theme. At the top, there are standard navigation icons: back, forward, search, and refresh. Below the toolbar is a code editor window containing PL/SQL code. The code is numbered from 92 to 116. It declares variables for employee, cashier, and table IDs, begins a transaction, inserts data into three tables, and handles exceptions by logging errors and rolling back the transaction if an error occurs. The code ends with a COMMIT and a DBMS\_OUTPUT message indicating the transaction is complete.

```
92  DECLARE
93    v_EMPLOYEE_ID NUMBER := 1;
94    v_CASHIER_ID NUMBER := 1;
95    v_TABLE_ID NUMBER := 1;
96  BEGIN
97    SAVEPOINT start_tran;
98    BEGIN
99      INSERT INTO EMPLOYEE (EMPLOYEE_ID, EMPLOYEE_NAME, EMPLOYEE_POS, EMPLOYEE_SALARY)
100     VALUES (v_EMPLOYEE_ID, 'John Doe', 'Manager', 5000);
101
102    INSERT INTO CASHIER (CASHIER_ID, CASHIER_NAME, CASHIER AGE, EMPLOYEE_ID)
103     VALUES (v_CASHIER_ID, 'Jane Smith', 25, v_EMPLOYEE_ID);
104
105    INSERT INTO TABLE_1 (TABLE_ID, TABLE_NUMBER, TABLE_CAPACITY)
106     VALUES (v_TABLE_ID, '3', 30);
107
108    COMMIT;
109    DBMS_OUTPUT.PUT_LINE('Transaction complete.');
110  EXCEPTION
111    WHEN OTHERS THEN
112      DBMS_OUTPUT.PUT_LINE('Error occurred: ' || SQLERRM);
113      ROLLBACK TO start_tran;
114      DBMS_OUTPUT.PUT_LINE('Transaction rolled back.');
115    END;
116  END;
```

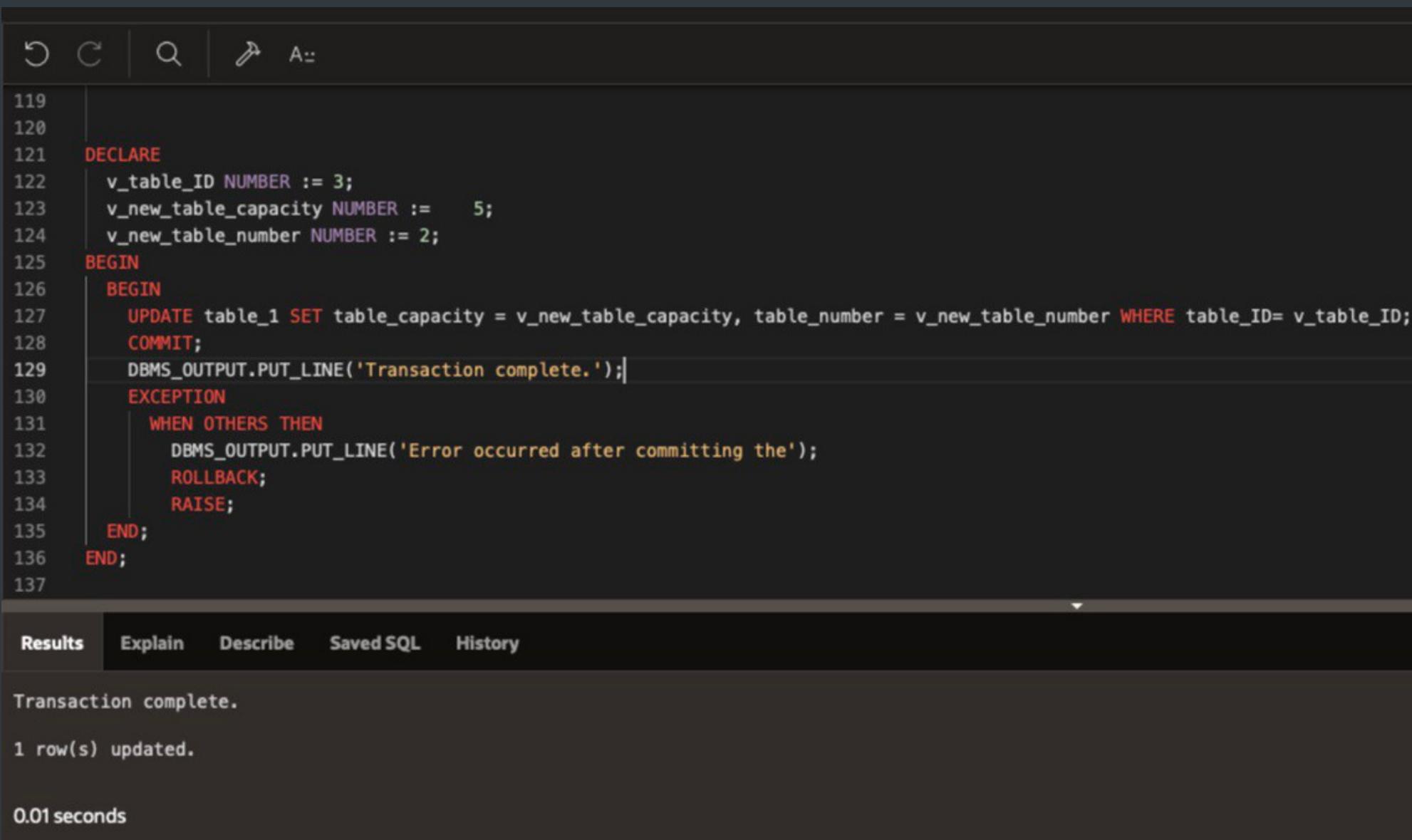
Below the code editor is a navigation bar with tabs: Results (which is selected), Explain, Describe, Saved SQL, and History.

The results pane displays the following output:

```
Error occurred: ORA-00001: unique constraint (WKSP_SDU.SYS_C00135115113) violated
Transaction rolled back.

1 row(s) inserted.
```

# Transaction 2



The screenshot shows a SQL developer interface with the following details:

- Code Area:** Displays PL/SQL code for Transaction 2. The code includes declarations for variables `v_table_ID`, `v_new_table_capacity`, and `v_new_table_number`, and performs an update on `table_1` followed by a commit. It also handles exceptions, specifically for errors occurring after committing.
- Status Bar:** Shows the status "Transaction complete." indicating the transaction has been committed.
- Results Tab:** Shows the output of the transaction: "1 row(s) updated." and a execution time of "0.01 seconds".
- Toolbar:** Includes icons for Undo, Redo, Find, and Paste.

```
declare
w_id waiter.waiter_id%type:=204;
w_age waiter.waiter_age%type;
w_name waiter.waiter_name%type;
| wboutage EXCEPTION;
BEGIN
SELECT waiter_name, waiter_age INTO w_name, w_age
| FROM waiter
| WHERE waiter_id = w_id;
IF w_age<18 then
| RAISE wboutage;
ELSE
| DBMS_OUTPUT.PUT_LINE ('Name: '|| w_name);
| DBMS_OUTPUT.PUT_LINE ('Age: ' || w_age);
END IF;

EXCEPTION
WHEN wboutage THEN
| dbms_output.put_line('Will be fired!');
WHEN no_data_found THEN
| dbms_output.put_line('No such waiter!');
WHEN others THEN
| dbms_output.put_line('Error!');
END;
```

# Exception 1

No such waiter!  
Statement processed.

0.06 seconds

```
1 declare
2   c_id  cashier.cashier_id%type:=204;
3   c_age cashier.cashier_age%type;
4   c_name cashier.cashier_name%type;
5   | cboutage EXCEPTION;
6 BEGIN
7   SELECT  cashier_name, cashier_age INTO  c_name, c_age
8   |   | FROM cashier
9   |   | WHERE cashier_id = c_id;
10  | IF c_age<18 then
11  |   | RAISE cboutage;
12  | ELSE
13  |   | DBMS_OUTPUT.PUT_LINE ('Name: ' || c_name);
14  |   | DBMS_OUTPUT.PUT_LINE ('Age: ' || c_age);
15 END IF;
```

```
EXCEPTION
  WHEN cboutage THEN
    dbms_output.put_line('Will be fired!');
  WHEN no_data_found THEN
    dbms_output.put_line('No such cashier!');
  WHEN others THEN
    dbms_output.put_line('Error!');
END;
```

# Exception 2

# Will be fired!

**Statement processed.**

0.00 seconds

```

declare
  e_id order_1.order_id%type:=58;
  e_status order_1.order_status%type;
  e_amount bill.total_amount%type;
  invalidorder exception;
begin
  select order_id, order_status, total_amount into e_id, e_status,e_amount
  from order_1, bill
  where
  order_1.client_id = bill.client_id
    AND order_1.order_id = e_id;
  if e_amount<=0 then
    raise invalidorder;
  else
    dbms_output.put_line('Order status: '||e_status);
  end if;

exception
  when invalidorder then
    dbms_output.put_line('Error!');
  when no_data_found then
    dbms_output.put_line('No such order');
  when others then
    dbms_output.put_line('Error');
end;

```

## Exception 3

Error!

Statement processed.

0.00 seconds

Order status: runout

Statement processed.

0.07 seconds

**Everything is fully explained in the video.**