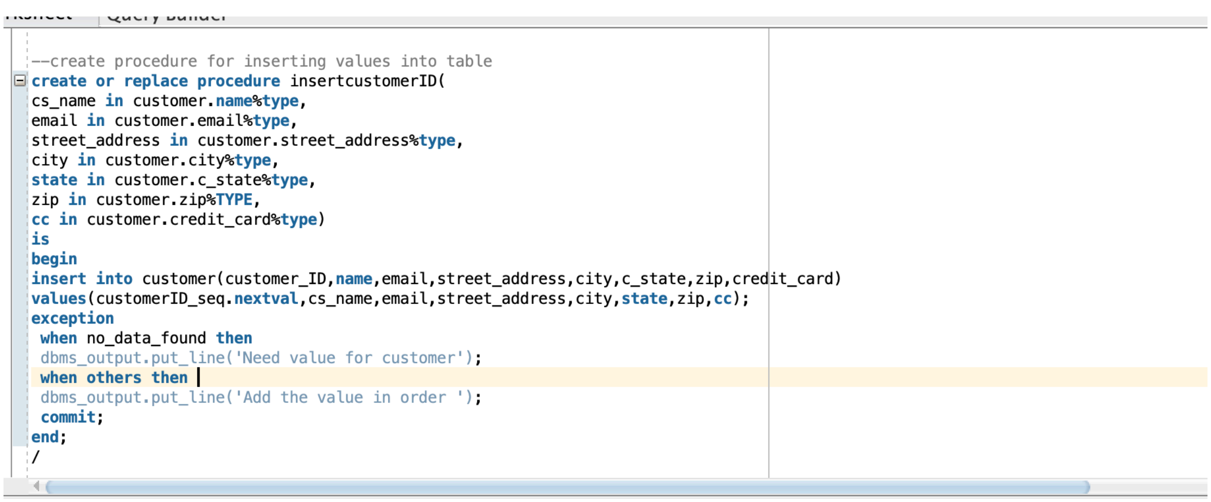
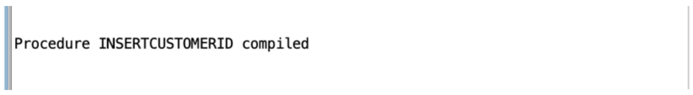
Rohan Salvi

1. Compiling procedure

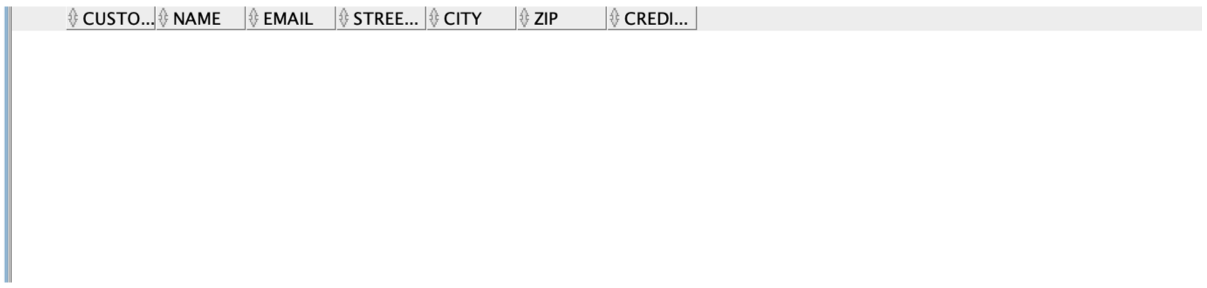


```
--create procedure for inserting values into table
create or replace procedure insertcustomerID(
cs_name in customer.name%type,
email in customer.email%type,
street_address in customer.street_address%type,
city in customer.city%type,
state in customer.c_state%type,
zip in customer.zip%TYPE,
cc in customer.credit_card%type)
is
begin
insert into customer(customer_ID,name,email,street_address,city,c_state,zip,credit_card)
values(customerID_seq.nextval,cs_name,email,street_address,city,state,zip,cc);
exception
 when no_data_found then
 dbms_output.put_line('Need value for customer');
 when others then
 dbms_output.put_line('Add the value in order ');
 commit;
end;
/
```



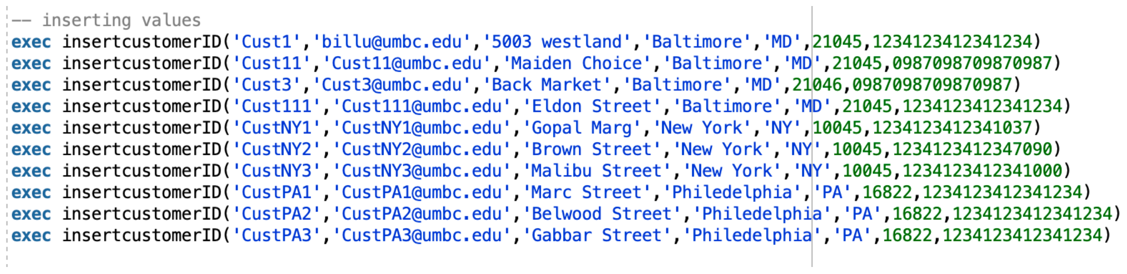
```
Procedure INSERTCUSTOMERID compiled
```

2. Select * from customer;



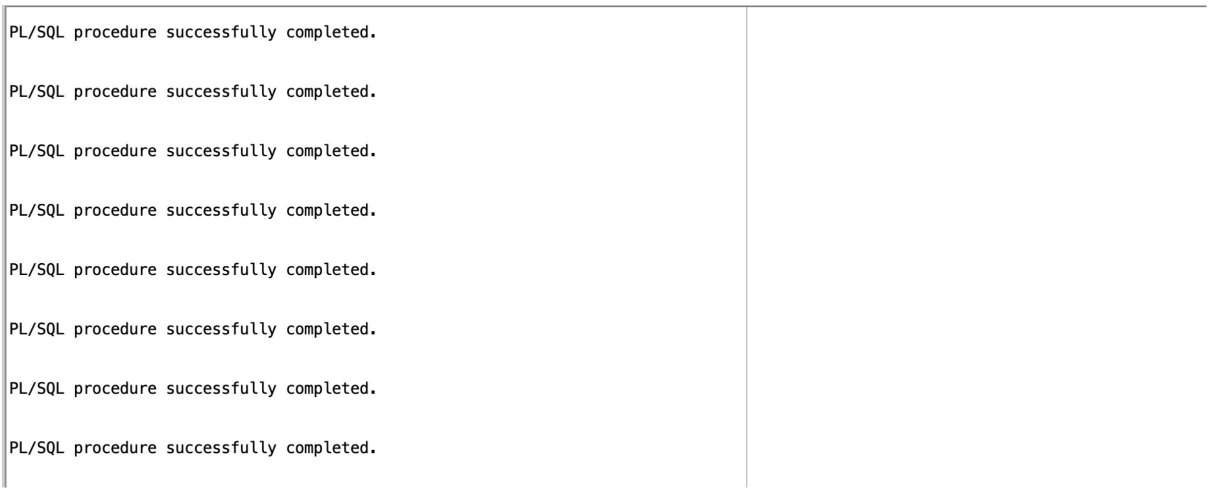
CUSTO...	NAME	EMAIL	STREE...	CITY	ZIP	CREDI...

3. Values to be inserted with the SQL query



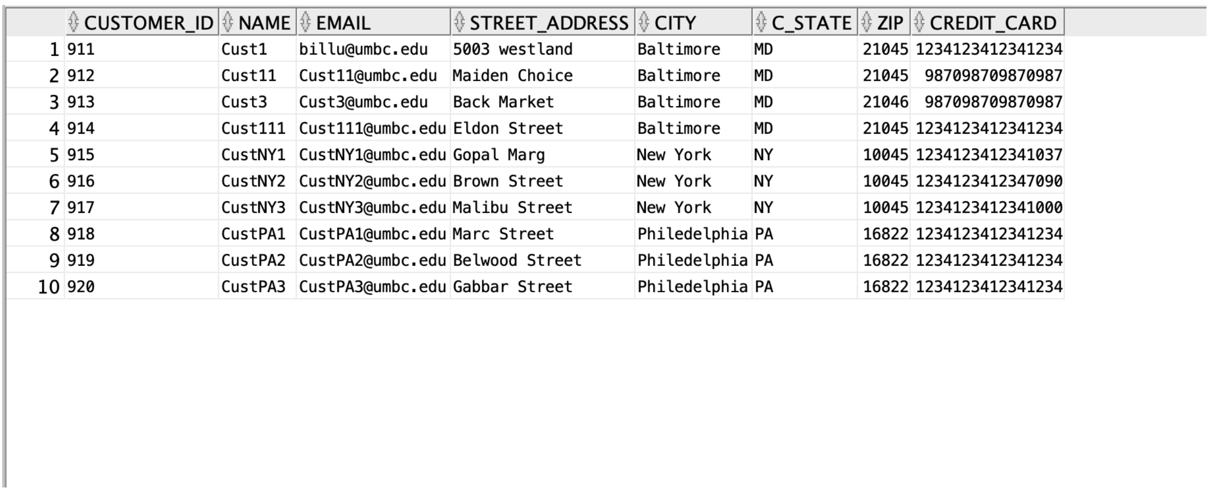
```
-- inserting values
exec insertcustomerID('Cust1','billu@umbc.edu','5003 westland','Baltimore','MD',21045,1234123412341234)
exec insertcustomerID('Cust11','Cust11@umbc.edu','Maiden Choice','Baltimore','MD',21045,0987098709870987)
exec insertcustomerID('Cust3','Cust3@umbc.edu','Back Market','Baltimore','MD',21046,0987098709870987)
exec insertcustomerID('Cust111','Cust111@umbc.edu','Eldon Street','Baltimore','MD',21045,1234123412341234)
exec insertcustomerID('CustNY1','CustNY1@umbc.edu','Gopal Marg','New York','NY',10045,1234123412341037)
exec insertcustomerID('CustNY2','CustNY2@umbc.edu','Brown Street','New York','NY',10045,1234123412347090)
exec insertcustomerID('CustNY3','CustNY3@umbc.edu','Malibu Street','New York','NY',10045,1234123412341000)
exec insertcustomerID('CustPA1','CustPA1@umbc.edu','Marc Street','Philedelphia','PA',16822,1234123412341234)
exec insertcustomerID('CustPA2','CustPA2@umbc.edu','Belwood Street','Philedelphia','PA',16822,1234123412341234)
exec insertcustomerID('CustPA3','CustPA3@umbc.edu','Gabbar Street','Philedelphia','PA',16822,1234123412341234)
```

4. Output for the above procedure for insertion



```
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

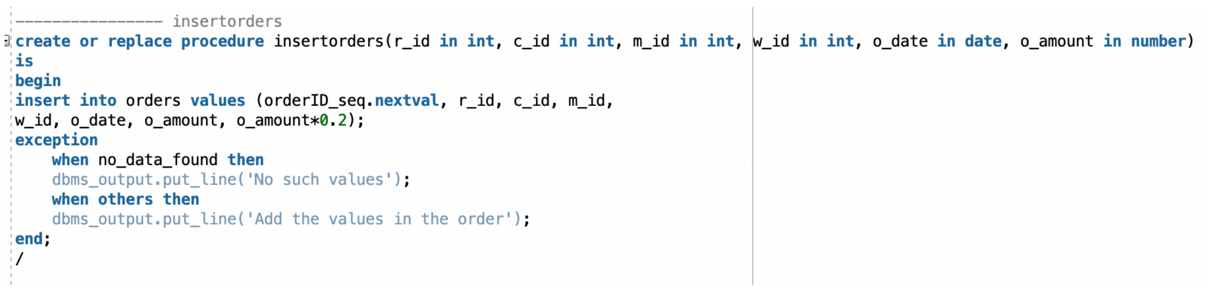
5. Select * from customer;



	CUSTOMER_ID	NAME	EMAIL	STREET_ADDRESS	CITY	C_STATE	ZIP	CREDIT_CARD
1	911	Cust1	billu@umbc.edu	5003 westland	Baltimore	MD	21045	1234123412341234
2	912	Cust11	Cust11@umbc.edu	Maiden Choice	Baltimore	MD	21045	987098709870987
3	913	Cust3	Cust3@umbc.edu	Back Market	Baltimore	MD	21046	987098709870987
4	914	Cust111	Cust111@umbc.edu	Eldon Street	Baltimore	MD	21045	1234123412341234
5	915	CustNY1	CustNY1@umbc.edu	Gopal Marg	New York	NY	10045	1234123412341037
6	916	CustNY2	CustNY2@umbc.edu	Brown Street	New York	NY	10045	1234123412347090
7	917	CustNY3	CustNY3@umbc.edu	Malibu Street	New York	NY	10045	1234123412341000
8	918	CustPA1	CustPA1@umbc.edu	Marc Street	Philedelphia	PA	16822	1234123412341234
9	919	CustPA2	CustPA2@umbc.edu	Belwood Street	Philedelphia	PA	16822	1234123412341234
10	920	CustPA3	CustPA3@umbc.edu	Gabbar Street	Philedelphia	PA	16822	1234123412341234

Procedure for insert orders

1. Code

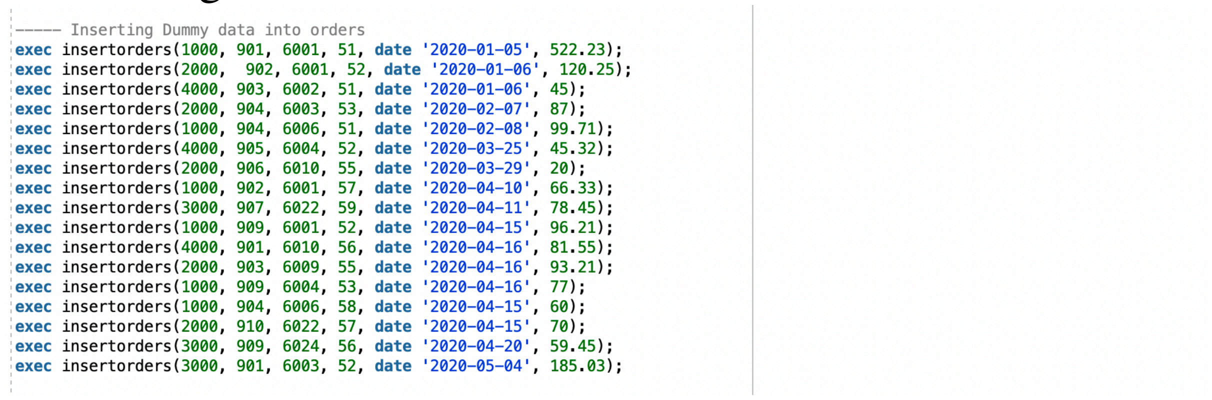


```
---------------- insertorders
create or replace procedure insertorders(r_id in int, c_id in int, m_id in int, w_id in int, o_date in date, o_amount in number)
is
begin
insert into orders values (orderID_seq.nextval, r_id, c_id, m_id,
w_id, o_date, o_amount, o_amount*0.2);
exception
    when no_data_found then
    dbms_output.put_line('No such values');
    when others then
    dbms_output.put_line('Add the values in the order');
end;
/
```

2. Procedure compiled

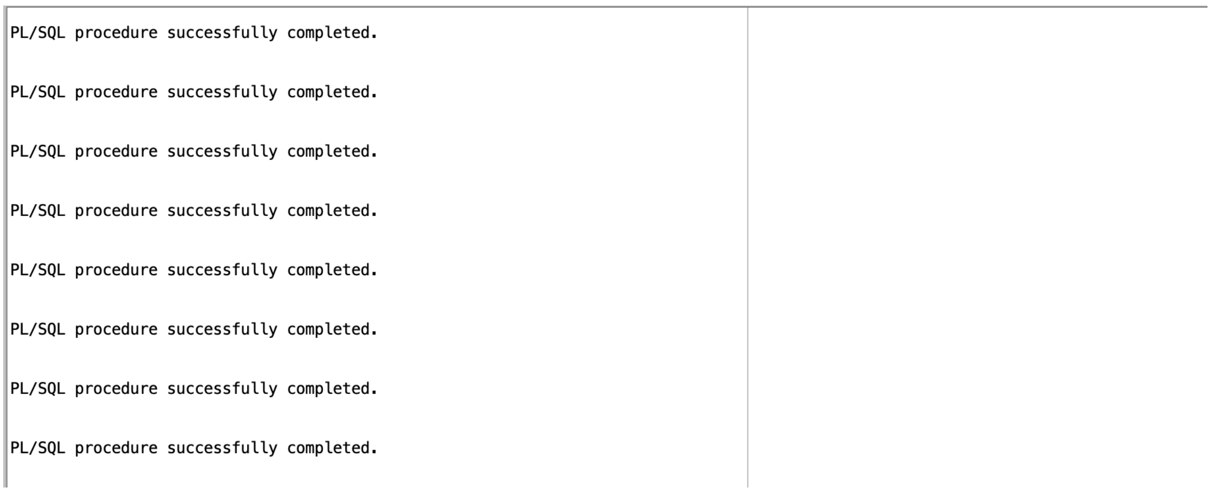
```
Procedure INSERTORDERS compiled
```

3. Inserting values



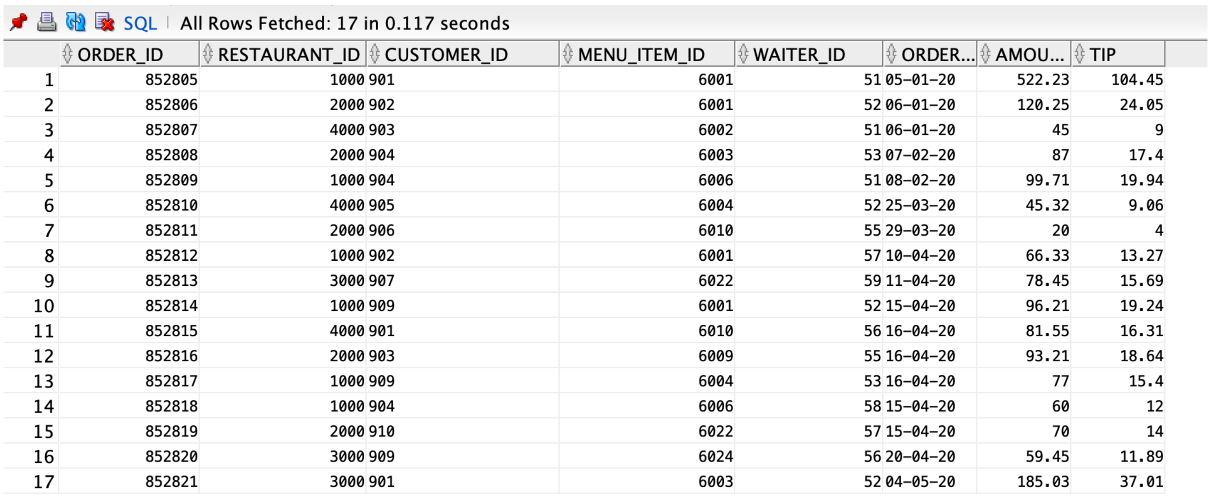
```
----- Inserting Dummy data into orders
exec insertorders(1000, 901, 6001, 51, date '2020-01-05', 522.23);
exec insertorders(2000,  902, 6001, 52, date '2020-01-06', 120.25);
exec insertorders(4000, 903, 6002, 51, date '2020-01-06', 45);
exec insertorders(2000, 904, 6003, 53, date '2020-02-07', 87);
exec insertorders(1000, 904, 6006, 51, date '2020-02-08', 99.71);
exec insertorders(4000, 905, 6004, 52, date '2020-03-25', 45.32);
exec insertorders(2000, 906, 6010, 55, date '2020-03-29', 20);
exec insertorders(1000, 902, 6001, 57, date '2020-04-10', 66.33);
exec insertorders(3000, 907, 6022, 59, date '2020-04-11', 78.45);
exec insertorders(1000, 909, 6001, 52, date '2020-04-15', 96.21);
exec insertorders(4000, 901, 6010, 56, date '2020-04-16', 81.55);
exec insertorders(2000, 903, 6009, 55, date '2020-04-16', 93.21);
exec insertorders(1000, 909, 6004, 53, date '2020-04-16', 77);
exec insertorders(1000, 904, 6006, 58, date '2020-04-15', 60);
exec insertorders(2000, 910, 6022, 57, date '2020-04-15', 70);
exec insertorders(3000, 909, 6024, 56, date '2020-04-20', 59.45);
exec insertorders(3000, 901, 6003, 52, date '2020-05-04', 185.03);
```

4. Output for the above procedure for insertion



```
PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.
```

5.Select * from orders;

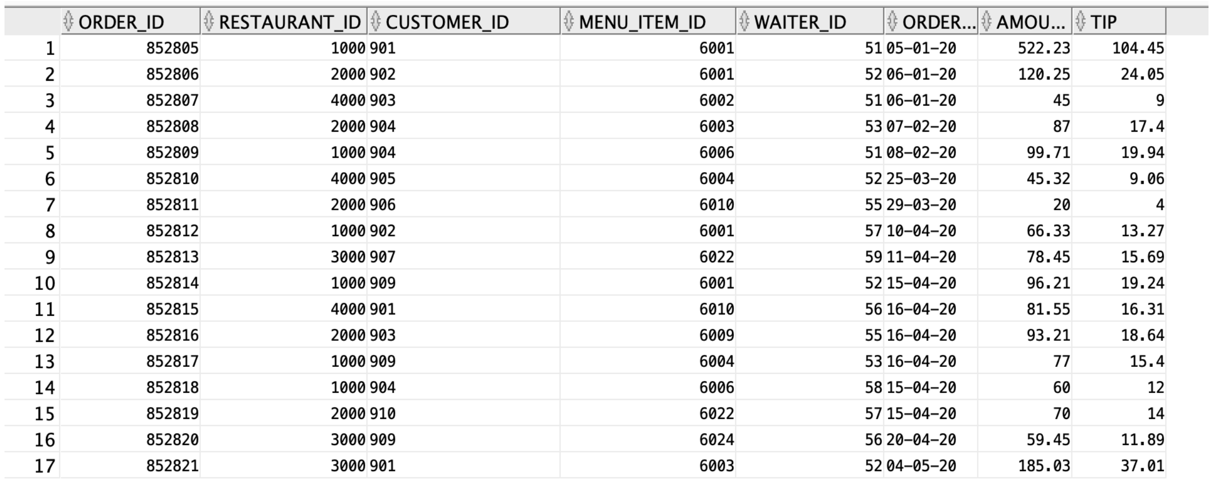


All Rows Fetched: 17 in 0.117 seconds

	ORDER_ID	RESTAURANT_ID	CUSTOMER_ID	MENU_ITEM_ID	WAITER_ID	ORDER...	AMOU...	TIP
1	852805	1000	901	6001	51	05-01-20	522.23	104.45
2	852806	2000	902	6001	52	06-01-20	120.25	24.05
3	852807	4000	903	6002	51	06-01-20	45	9
4	852808	2000	904	6003	53	07-02-20	87	17.4
5	852809	1000	904	6006	51	08-02-20	99.71	19.94
6	852810	4000	905	6004	52	25-03-20	45.32	9.06
7	852811	2000	906	6010	55	29-03-20	20	4
8	852812	1000	902	6001	57	10-04-20	66.33	13.27
9	852813	3000	907	6022	59	11-04-20	78.45	15.69
10	852814	1000	909	6001	52	15-04-20	96.21	19.24
11	852815	4000	901	6010	56	16-04-20	81.55	16.31
12	852816	2000	903	6009	55	16-04-20	93.21	18.64
13	852817	1000	909	6004	53	16-04-20	77	15.4
14	852818	1000	904	6006	58	15-04-20	60	12
15	852819	2000	910	6022	57	15-04-20	70	14
16	852820	3000	909	6024	56	20-04-20	59.45	11.89
17	852821	3000	901	6003	52	04-05-20	185.03	37.01

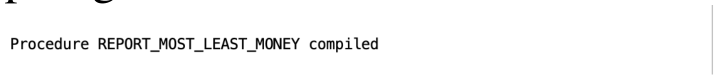
Procedure for report of Most(top 3) and least(bottom 3) money spent

Select * from orders;



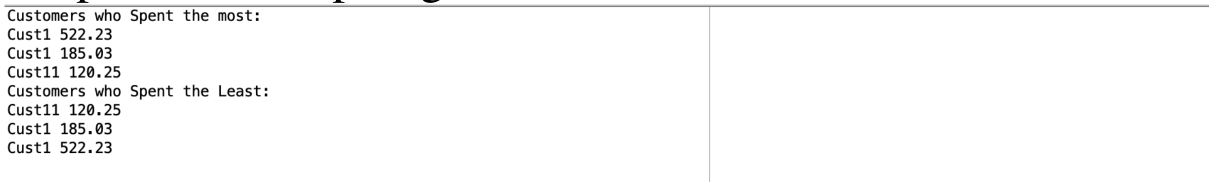
	ORDER_ID	RESTAURANT_ID	CUSTOMER_ID	MENU_ITEM_ID	WAITER_ID	ORDER...	AMOU...	TIP
1	852805	1000	901	6001	51	05-01-20	522.23	104.45
2	852806	2000	902	6001	52	06-01-20	120.25	24.05
3	852807	4000	903	6002	51	06-01-20	45	9
4	852808	2000	904	6003	53	07-02-20	87	17.4
5	852809	1000	904	6006	51	08-02-20	99.71	19.94
6	852810	4000	905	6004	52	25-03-20	45.32	9.06
7	852811	2000	906	6010	55	29-03-20	20	4
8	852812	1000	902	6001	57	10-04-20	66.33	13.27
9	852813	3000	907	6022	59	11-04-20	78.45	15.69
10	852814	1000	909	6001	52	15-04-20	96.21	19.24
11	852815	4000	901	6010	56	16-04-20	81.55	16.31
12	852816	2000	903	6009	55	16-04-20	93.21	18.64
13	852817	1000	909	6004	53	16-04-20	77	15.4
14	852818	1000	904	6006	58	15-04-20	60	12
15	852819	2000	910	6022	57	15-04-20	70	14
16	852820	3000	909	6024	56	20-04-20	59.45	11.89
17	852821	3000	901	6003	52	04-05-20	185.03	37.01

Compiling

1. 

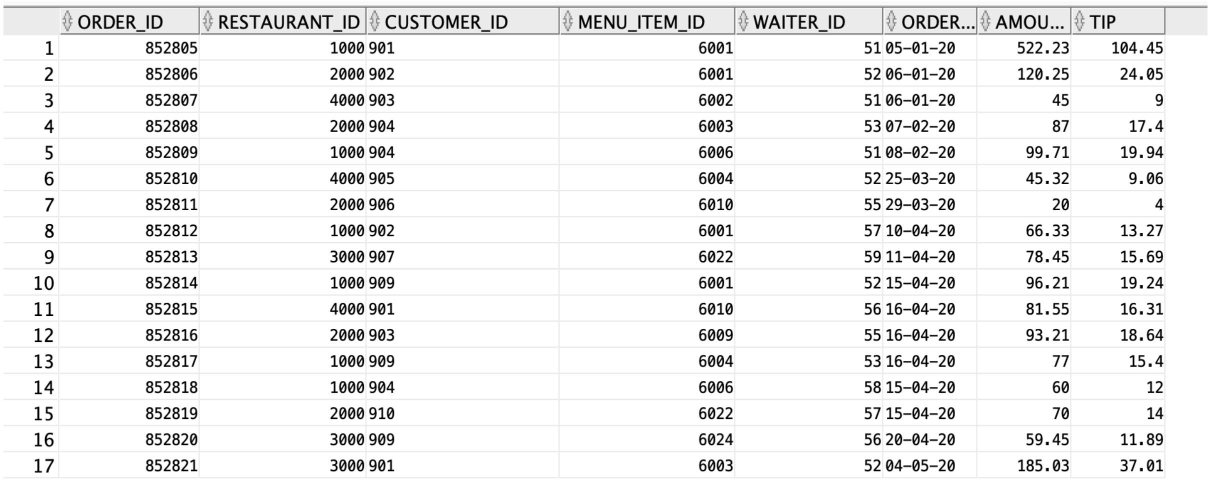
```
Procedure REPORT_MOST_LEAST_MONEY compiled
```

2. Output after compiling



```
Customers who Spent the most:
Cust1 522.23
Cust1 185.03
Cust11 120.25
Customers who Spent the Least:
Cust11 120.25
Cust1 185.03
Cust1 522.23
```

Procedure for States of generous customers.
Select * from orders;



	ORDER_ID	RESTAURANT_ID	CUSTOMER_ID	MENU_ITEM_ID	WAITER_ID	ORDER...	AMOU...	TIP
1	852805	1000	901	6001	51	05-01-20	522.23	104.45
2	852806	2000	902	6001	52	06-01-20	120.25	24.05
3	852807	4000	903	6002	51	06-01-20	45	9
4	852808	2000	904	6003	53	07-02-20	87	17.4
5	852809	1000	904	6006	51	08-02-20	99.71	19.94
6	852810	4000	905	6004	52	25-03-20	45.32	9.06
7	852811	2000	906	6010	55	29-03-20	20	4
8	852812	1000	902	6001	57	10-04-20	66.33	13.27
9	852813	3000	907	6022	59	11-04-20	78.45	15.69
10	852814	1000	909	6001	52	15-04-20	96.21	19.24
11	852815	4000	901	6010	56	16-04-20	81.55	16.31
12	852816	2000	903	6009	55	16-04-20	93.21	18.64
13	852817	1000	909	6004	53	16-04-20	77	15.4
14	852818	1000	904	6006	58	15-04-20	60	12
15	852819	2000	910	6022	57	15-04-20	70	14
16	852820	3000	909	6024	56	20-04-20	59.45	11.89
17	852821	3000	901	6003	52	04-05-20	185.03	37.01

1. Compiling

```
Procedure REPORT_STATE_OF_GENROUS_CS compiled
```

2. Output after compiling

```
States of generous customers:
MD 1360.31
PA 302.66
NY 143.77


PL/SQL procedure successfully completed.
```