AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

Course Title
INTRODUCTION TO DATABASE

Fall, 23-24
Section: H

TITLE
Restaurant Management System

Supervised By
Razuan Karim

Submitted By: Group no: 03.

Name	ID
DIBAJIT ROY	22-48569-3
MAHMUDUL HASAN MARUF	22-48580-3
MD MARUF HASAN CHOWDHURY	22-48582-3
MIRAT HASAN NILOY	22-48545-3

TABLE OF CONTENTS

November and December 2023				
Item	**NOV (1-16)**	**NOV (17- 30)**	**DEC (1-14)**	**DEC (15-20)**
Introduction,Case Study ,E-R diagram,				
Normalaization, Finalization, Table Creation				
Data Insertion, Simple Query Aggregate Query, Single -row, Multiple-row subquery Joining,				
View, Database Connection, , Conclusion				

Introduction: The Restaurant Management System (RMS) presented in this project is acomprehensive solution designed to meet the evolving needs of the dynamic restaurant industry. With a focus on leveraging technology to enhance operational efficiency, the RMS aims to revolutionize how restaurants manage orders, inventory, employees, and customer interactions. Key features include streamlined order processing, efficient inventory management, employee and customer relationship management (CRM). By embracing the capabilities of the RMS, restaurant owners andstaff can expect improved productivity, reduced manual errors, and an enhanced overall dining experience for their customers in the competitive and fast-paced restaurant landscape.

Case Study: A **restaurant** has a unique restaurant ID(**R_ID**)**,** restaurant name(**R_Name**), contact number(**R_contact**) and (**Address**). Many customers can go to one restaurant. A **customer** has unique customers ID(**Cus_ID**), customers Name (**Cus_Name**) and customer contact number (**Cus_contact**). A customer places many orders, the waiter take order. One waitercan take many orders. Then the waiter will serve the customer their order. One waiter can serve more than one customer. A **waiter** has unique waiter ID(**W_ID**) and waiter name(**W_Name**). An **order** has unique order number (**Order_NO**), Number of items (**No_Items**) and order time (**Ord_Time**). Then the order is prepared by the chef. Many chefs prepared many orders. Moreover, a **chef** has unique ID (**Chef_ID**) and chef name (**Chef_Name**).An order contains **food** (one order contains many food) which has unique food number (**Food_NO**), (**Quantity**), (**Price**), and (**Description**). Customers pay bills (one customers can pay more than one bill). The **bills** contain unique bill number(**B_NO**), price(**B_Price**), (**Order_Detail**), and (**vat**). A restaurant has one manager,**Manager** has unique manager ID(**Man_ID**) and manager name(**Man_Name**).One manager can manage many waiter.

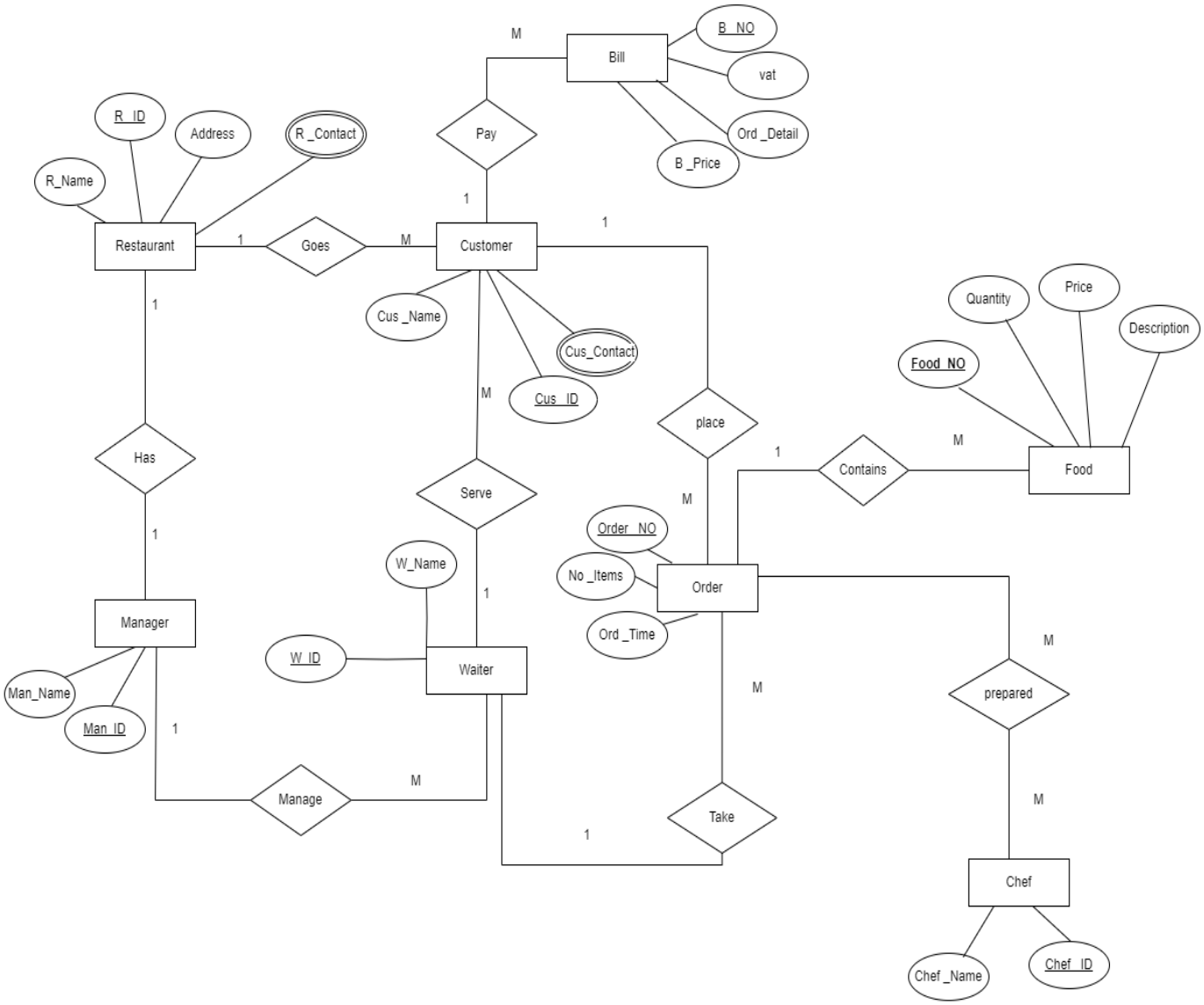


Fig 1: E-R Diagram for Restaurant Management System

Normalization:

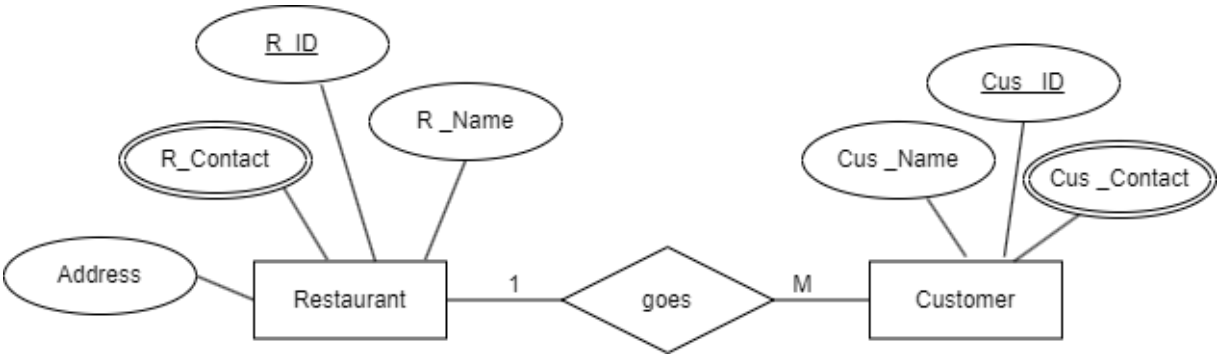


Fig 2: Relation between Restaurant and Customer

Goes:

UNF:(**Cus _ID**, Cus _Name, Cus _Contact**, R_ID**, R _Name, Address, R _Contact)

1NF:

Customer: **Cus _ID**, Cus_ Name, Cus_Contact

Restaurant: **R _ID**, R_Name, Address, R _Contact

2NF:

Customer: **Cus _ID**, Cus _Name, Cus_Contact,R_ID(FK)

Restaurant: **R_ID**, R _Name, Address, R_Contact

3NF:

Restaurant: **R_ID**, R_Name, R _Contact, R_ID(FK)

Restaurant INFO: R _Name, Address

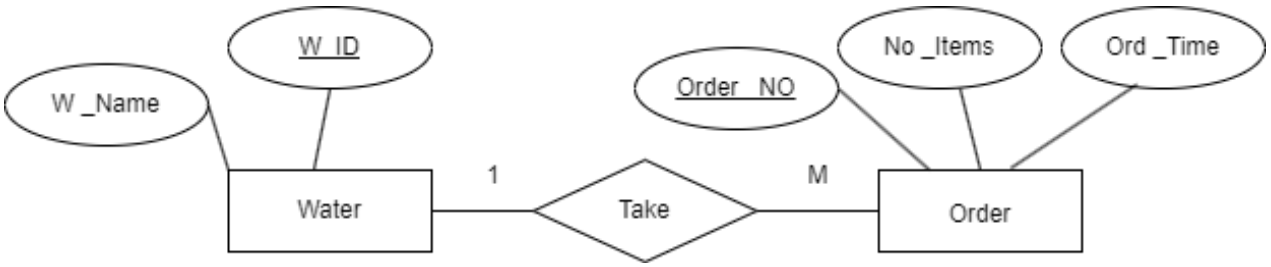


Fig 3: Relation between Waiter and Order

Take:

UNF:(**W_ID**, W _Name, Order _NO, No _Items, Ord _Time)
1NF:
Waiter: **W_ID**, W_Name
Order: **Order_NO**, No _Items, Ord _Time
2NF:
Waiter: **W_ID**, W _Name, Order _No (FK)
Order: **Order_NO**, No _Items, Ord _Time
3NF:
Waiter: **W_ID**, W _Name, Order _No (FK)
Order: **Order_NO**, No _Items
OrderINFO: No _Items, Ord _Time

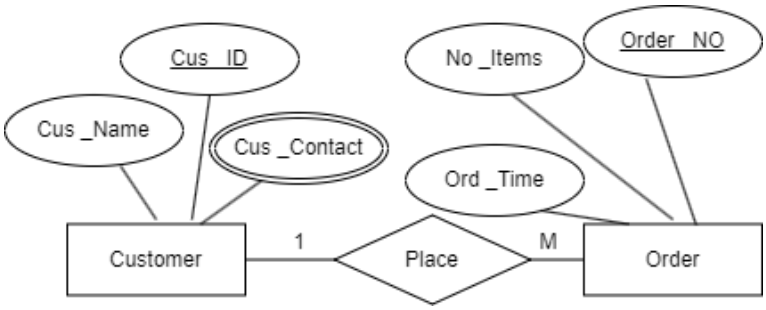


Fig 4: Relation between Customer and Order

Place:

UNF: **Cus_ID**, Cus _Name, Cus _Contact, **Order_NO**, No _Items, Ord _Time
1NF:
Customer: **Cus_ID**, Cus _Name, Cus _Contact
Order: **Order_NO**, No _Items, Ord _Time
2NF:
Customer: **Cus_ID**, Cus _Name, Cus_Contact
Order: **Order_NO**, No_Items, Ord_Time, Cus_ID(FK)
3NF:
Customer: **Cus_ID**, Cus_Name, Cus_Contact
Order: **Order_NO**, No_Items, Cus_ID(FK)
OrderINFO: No_Items, Ord_Time

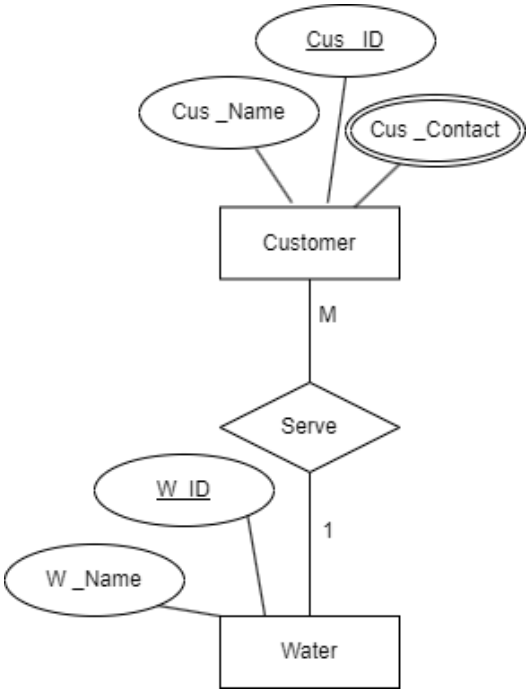


Fig 5: Relation between Customer and Waiter

Serve:

UNF:(Customer: **Cus_ID**, Cus _Name, Cus _Contact, **W_ID**, W _Name)

1NF:

Customer: **Cus_ID**, Cus _Name, Cus _Contact

Waiter: **W_ID**, W_Name

2NF:

Customer: **Cus_ID**, Cus _Name, Cus _Contact

Waiter: **W_ID**, W _Name, Cus _ID(FK)

3NF:

Same As 2NF

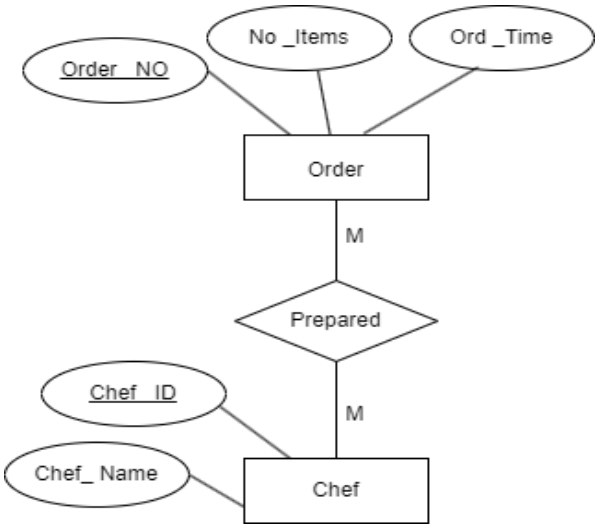


Fig 6: Relation between Order and Chef

Prepared

UNF:(**Chef _ID**, Chef_ Name, Order _NO, No _Items, Ord _Time)
1NF:
Chef: **Chef _ID**, Chef _Name
Order: Order _NO, No _Items, Ord _Time
2NF:
Chef: **Chef _ID**, Chef _Name, Order _NO(FK)
Order: **Order _NO**, No _Items, Ord _Time
3NF:
Chef: **Chef _ID**, Chef _Name, Order _NO(FK)
Order**: Order _NO**, No _Items
OrderINFO: No _Items, Ord _Time

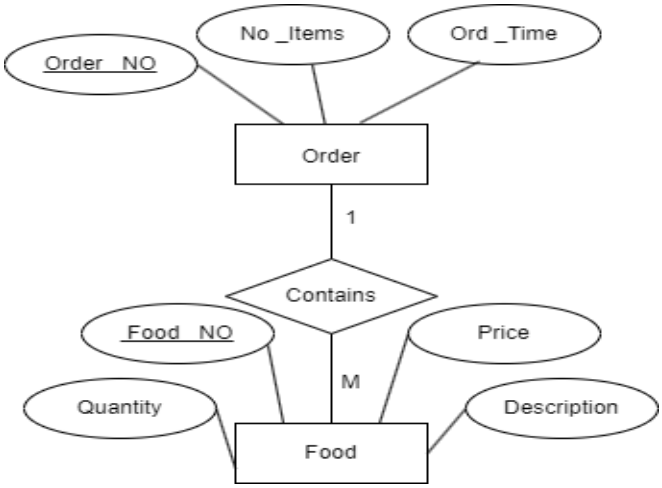


Fig 7: Relation between Order and Food

Contains

UNF:(**Order _NO**, No _Items, Ord _Time, **Food _NO**, Quantity, Price)

1NF:
Order: **Order _NO**, No _Items, Ord _Time
Food: **Food _NO**, Quantity, Price,
2NF:
Order: **Order_NO**, No _Items, Ord _Time
Food: **Food_NO**, Quantity, Price, Description, Order _NO(FK)
3NF:
Order: **Order_NO**, No_Items

OrderINFO: No _Items, Ord _Time
Food: **Food_NO**, Quantity, Description, Order_NO(FK)
FoodDetails: Quantity,Price, Food_NO(FK)

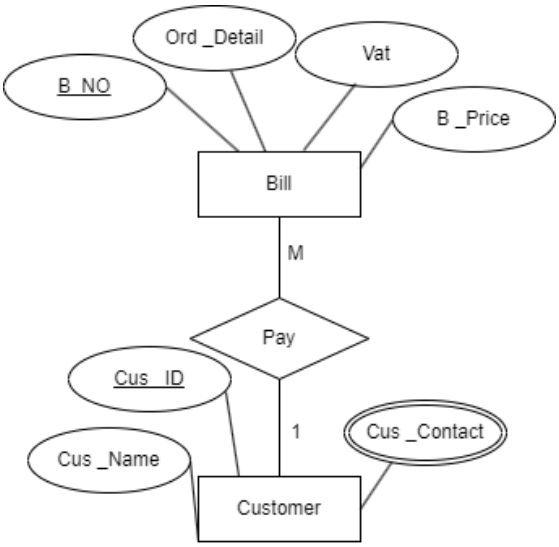


Fig 8: Relation between Customer and Bill

Pay

UNF:(Customer: **Cus_ID**, Cus _Name, Cus _Contact, **B_NO**, Vat, Ord _Detail, B _Price)
1NF:
Customer: **Cus_ID**, Cus _Name, Cus _Contact
Bill: **B_NO**, Vat, Ord _Detail, B_Price
2NF:
Customer: **Cus_ID**, Cus _Name, Cus _Contact
Bill: **B_NO**, Vat, Ord _Detail, B _Price, Cus _ID(FK)
3NF:
Customer: **Cus_ID**, Cus _Name, Cus _Contact
Bill: **B_NO**, Ord_Detail, B_Prcie, Cus_ID(FK)
BillDescription: B_Price, Vat

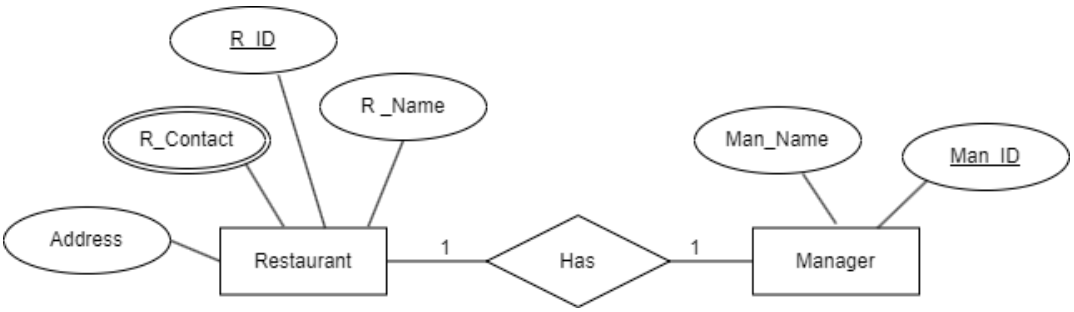


Fig 9: Relation between Restaurant and Manager

Has

UNF:(**R_ID**, R _Name, Address, R_Contact, **Man_ID**, Man_Name)
1NF:
Restaurant: **R_ID**, R_Name, Address, R_Contact
Manager: **Man_ID**, Man _Name
2NF:
Restaurant: **R_ID**, R_ Contact, Man_ID(FK)
Manager: **Man_ID**, Man_Name
Restaurant INFO: R _Name, Address

ManagerDetails: **R_ID**, Man_ID(FK)
3NF: Same as 2NF

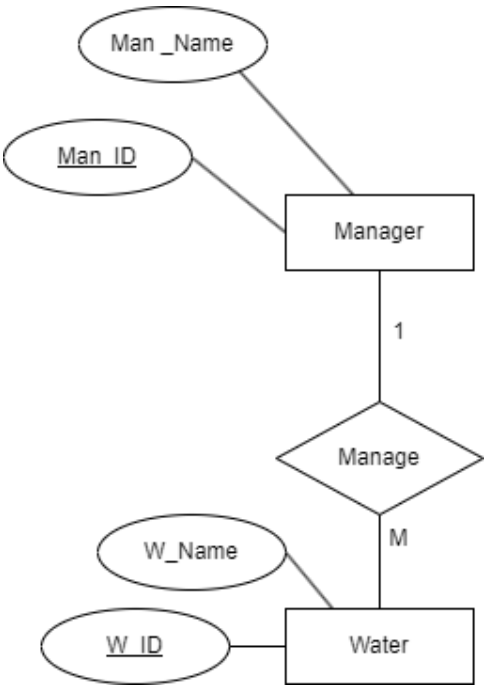


Fig 10: Relation between Manager and Waiter

Manage

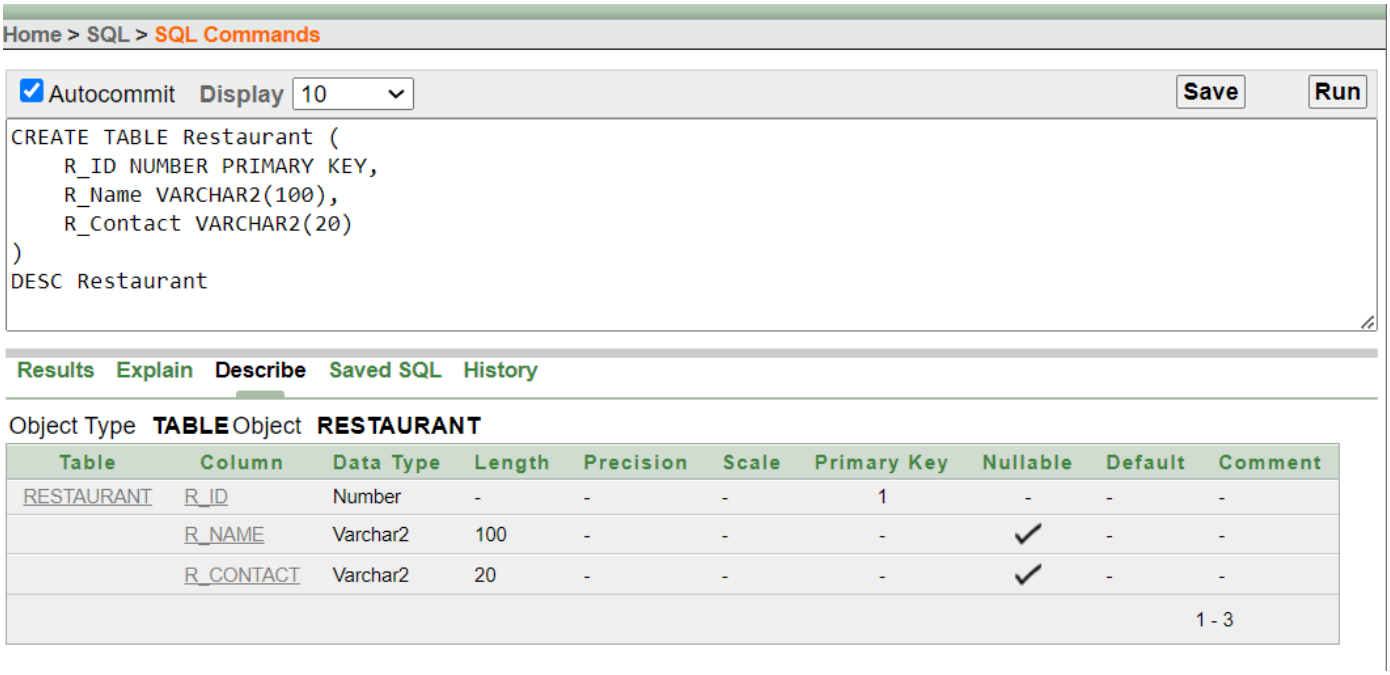
UNF:(**Man_ID**, Man_Name, **W_ID**, W_Name)
1NF:
Manager: **Man_ID**, Man _Name
Waiter: **W_ID**,W_Name
2NF:
Manager: **Man_ID**,Man_Name
Waiter: **W_ID**, W_Name, Man_ID(FK)
3NF: Same as 2NF

Finalization:

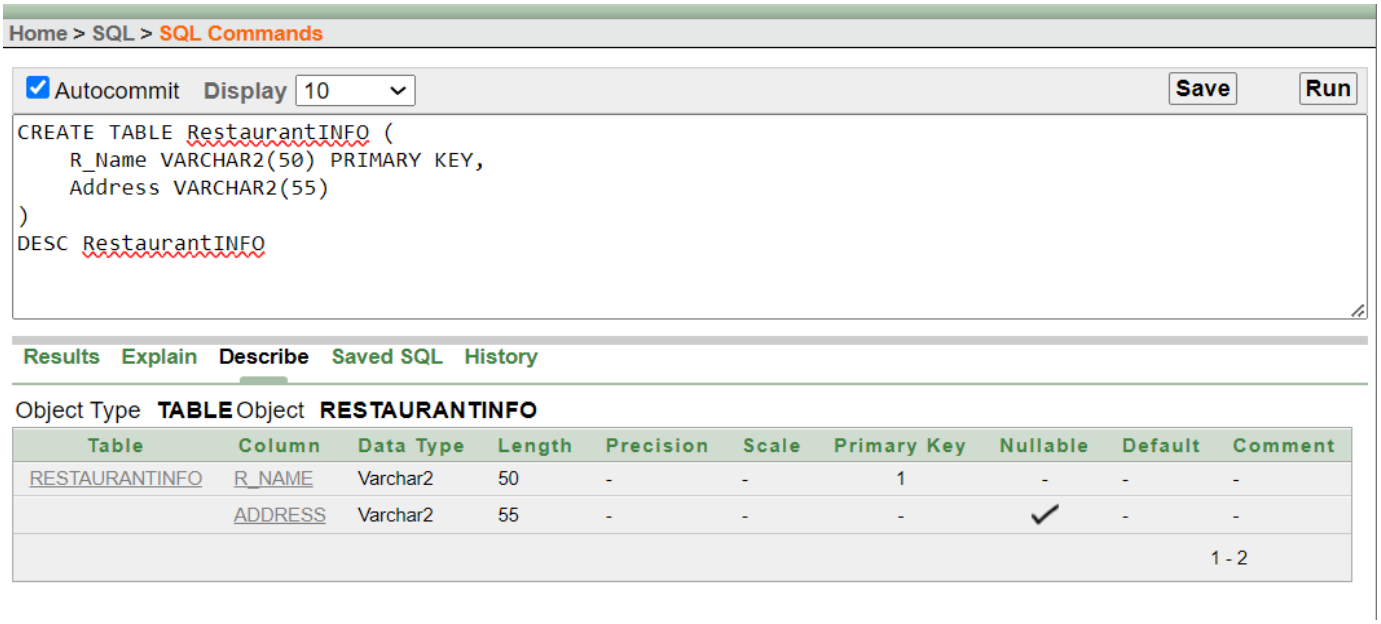
I. Restaurant: **R_ID**, R_Name, R _Contact
II. RestaurantINFO: R _Name , Address
III. Waiter**: W_ID**, W _Name, Order _No (FK)
IV. Order: **Order _NO**, No _Items
V. OrderINFO: No _Items , Ord _Time
VI. Customer: **Cus _ID**, Cus _Name, Cus _Contact, R_ID(FK)
VII. Order: **Order _NO**, No _Items, Cus _ID(FK)
~~VIII. OrderINFO: No _Items , Ord _Time~~
~~IX. Customer: **Cus _ID**, Cus _Name, Cus _Contact~~
X. Waiter**: W _ID**, W _Name, Cus _ID(FK)
XI. Chef: **Chef _ID**, Chef _Name, Order _NO(FK)
~~XII. Order: **Order _NO**, No _Items~~
~~XIII. OrderINFO: No _Items , Ord _Time~~
~~XIV. Order: **Order_NO**, No_Items~~
~~XV. OrderINFO: No _Items , Ord _Time~~
XVI. Food: **Food_NO**, Quantity, Description, Order_NO(FK)
XVII. FoodDetails: Quantity, Price, Food_NO(FK)
~~XVIII. Customer: **Cus _ID,** Cus _Name, Cus _Contact~~
XIX. Bill: **B_NO**, Ord_Detail, B_Prcie, Cus_ID(FK)
XX. BillDescription: B_Price,Vat
~~XXI. Restaurant: **R_ID**, R_Name, R_ Contact~~
XXII. Manager: **Man_ID**, Man_Name
~~XXIII. RestaurantINFO: R _Name , Address~~
XXIV. ManagerDetails: **R_ID**, Man_ID(FK)
~~XXV. Manager: **Man_ID**, Man_Name~~
XXVI. Waiter: **W_ID,** W_Name, Man_ID(FK)

Table Creation

1. Restaurant Table



2. RestaurantINFO Table



3. Customer Table

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
CREATE TABLE Customer (
    Cus_ID NUMBER PRIMARY KEY,
    Cus_Name VARCHAR2(100),
    Cus_Contact VARCHAR2(20),
    R_ID NUMBER,
    CONSTRAINT fk_Restaurant FOREIGN KEY (R_ID) REFERENCES Restaurant(R_ID)
)
DESC Customer
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **CUSTOMER**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Co
CUSTOMER	CUS_ID	Number	-	-	-	1	-	-	-
	CUS_NAME	Varchar2	100	-	-	-	✓	-	-
	CUS_CONTACT	Varchar2	20	-	-	-	✓	-	-
	R_ID	Number	-	-	-	-	✓	-	-

1 - 4

4. Waiter Table

Home > SQL > SQL Commands

Autocommit Display 15 Save Run

```
CREATE TABLE Waiter (
    W_ID NUMBER PRIMARY KEY,
    W_Name VARCHAR2(25),
    Cus_ID NUMBER,
    Order_No NUMBER,
    CONSTRAINT fk_Customer FOREIGN KEY (Cus_ID) REFERENCES Customer(Cus_ID),
    CONSTRAINT fk_Order FOREIGN KEY (Order_No) REFERENCES Orderr(Order_NO)
)

DESC Waiter
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **WAITER**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
WAITER	W_ID	Number	-	-	-	1	-	-	-
	W_NAME	Varchar2	25	-	-	-	✓	-	-
	CUS_ID	Number	-	-	-	-	✓	-	-
	ORDER_NO	Number	-	-	-	-	✓	-	-

1 - 4

5. Orderr Table

Home > SQL > SQL Commands

Autocommit Display 15 Save Run

```
CREATE TABLE Orderr(
    Order_NO NUMBER PRIMARY KEY,
    No_Items NUMBER
)

DESC Orderr
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **ORDERR**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORDERR	ORDER_NO	Number	-	-	-	1	-	-	-
	NO_ITEMS	Number	-	-	-	-	✓	-	-

1 - 2

6. OrderINFO Table

Home > SQL > SQL Commands

Autocommit Display 15 Save Run

```
CREATE TABLE OrderINFO (
    No_Items NUMBER PRIMARY KEY,
    Ord_Time TIMESTAMP
)

DESC OrderINFO
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **ORDERINFO**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ORDERINFO	NO_ITEMS	Number	-	-	-	1	-	-	-
	ORD_TIME	Timestamp(6)	11	-	6	-	✓	-	-

1 - 2

7. Food Table

Home > SQL > SQL Commands

Autocommit Display 15 Save Run

```
CREATE TABLE Food (
    Food_NO NUMBER PRIMARY KEY,
    Quantity NUMBER,
    Description VARCHAR2(255),
    Order_NO NUMBER,
    CONSTRAINT fk_Food_Order FOREIGN KEY (Order_NO) REFERENCES Orderr(Order_NO)
)


DESC Food
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **FOOD**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FOOD	FOOD_NO	Number	-	-	-	1	-	-	-
	QUANTITY	Number	-	-	-	-	✓	-	-
	DESCRIPTION	Varchar2	255	-	-	-	✓	-	-
	ORDER_NO	Number	-	-	-	-	✓	-	-

1 - 4

8. FoodDetails Table

Home > SQL > SQL Commands

Autocommit Display 15 Save Run

```
CREATE TABLE FoodDetails (
    Quantity NUMBER PRIMARY KEY,
    Price NUMBER
)


DESC FoodDetails
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **FOODDETAILS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
FOODDETAILS	QUANTITY	Number	-	-	-	1	-	-	-
	PRICE	Number	-	-	-	-	✓	-	-

1 - 2

9. Chef Table

Home > SQL > SQL Commands

Autocommit Display 15 Save Run

```
CREATE TABLE Chef (
    Chef_ID NUMBER PRIMARY KEY,
    Chef_Name VARCHAR2(100),
    Order_NO NUMBER,
    CONSTRAINT fk_Chef_Order FOREIGN KEY (Order_NO) REFERENCES Orderr(Order_NO)
)

DESC Chef
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **CHEF**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CHEF	CHEF_ID	Number	-	-	-	1	-	-	-
	CHEF_NAME	Varchar2	100	-	-	-	✓	-	-
	ORDER_NO	Number	-	-	-	-	✓	-	-

1 - 3

10. Bill Table

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
CREATE TABLE Bill (
    B_NO NUMBER PRIMARY KEY,
    B_Price NUMBER,
    Ord_Detail VARCHAR2(55),
    Cus_ID NUMBER,
    CONSTRAINT fk_Customer_Bill FOREIGN KEY (Cus_ID) REFERENCES Customer(Cus_ID)
)

DESC Bill
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **BILL**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BILL	B_NO	Number	-	-	-	1	-	-	-
	B_PRICE	Number	-	-	-	-	✓	-	-
	ORD_DETAIL	Varchar2	55	-	-	-	✓	-	-
	CUS_ID	Number	-	-	-	-	✓	-	-

1 - 4

11. BillDescription Table

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
CREATE TABLE BillDescription (
    Price NUMBER PRIMARY KEY,
    Vat VARCHAR2(15)
)


DESC BillDescription
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **BILLDESCRIPTION**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BILLDESCRIPTION	PRICE	Number	-	-	-	1	-	-	-
	VAT	Varchar2	15	-	-	-	✓	-	-

1 - 2

12. Manager Table

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
CREATE TABLE Manager (
    Man_ID NUMBER PRIMARY KEY,
    Man_Name VARCHAR2(50)
)


DESC Manager
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **MANAGER**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MANAGER	MAN_ID	Number	-	-	-	1	-	-	-
	MAN_NAME	Varchar2	50	-	-	-	✓	-	-

1 - 2

13. ManagerDetails Table

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
CREATE TABLE ManagerDetails (
    R_ID NUMBER PRIMARY KEY,
    Man_ID NUMBER,
    CONSTRAINT fk_Manager_Details FOREIGN KEY (Man_ID) REFERENCES Manager(Man_ID)
)

DESC ManagerDetails
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **MANAGERDETAILS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MANAGERDETAILS	R_ID	Number	-	-	-	1	-	-	-
	MAN_ID	Number	-	-	-	-	✓	-	-

1 - 2

14. WaiterManage Table

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
CREATE TABLE WaiterManage (
    W_ID NUMBER PRIMARY KEY,
    W_Name VARCHAR2(50),
    Man_ID NUMBER,
    CONSTRAINT fk_Manager_Waiter FOREIGN KEY (Man_ID) REFERENCES Manager(Man_ID)
)

DESC WaiterManage
```

Results Explain Describe Saved SQL History

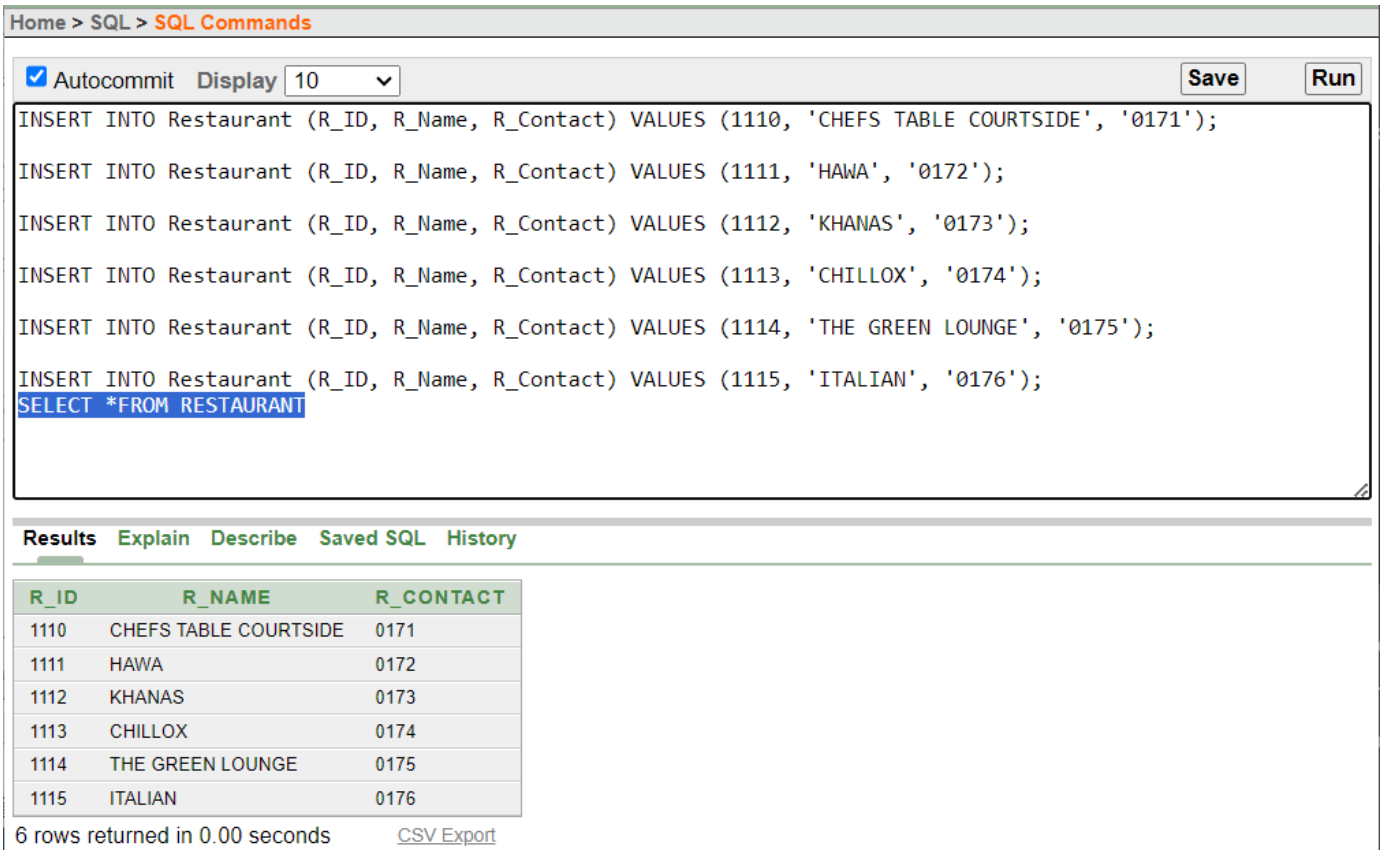
Object Type **TABLE** Object **WAITERMANAGE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
WAITERMANAGE	W_ID	Number	-	-	-	1	-	-	-
	W_NAME	Varchar2	50	-	-	-	✓	-	-
	MAN_ID	Number	-	-	-	-	✓	-	-

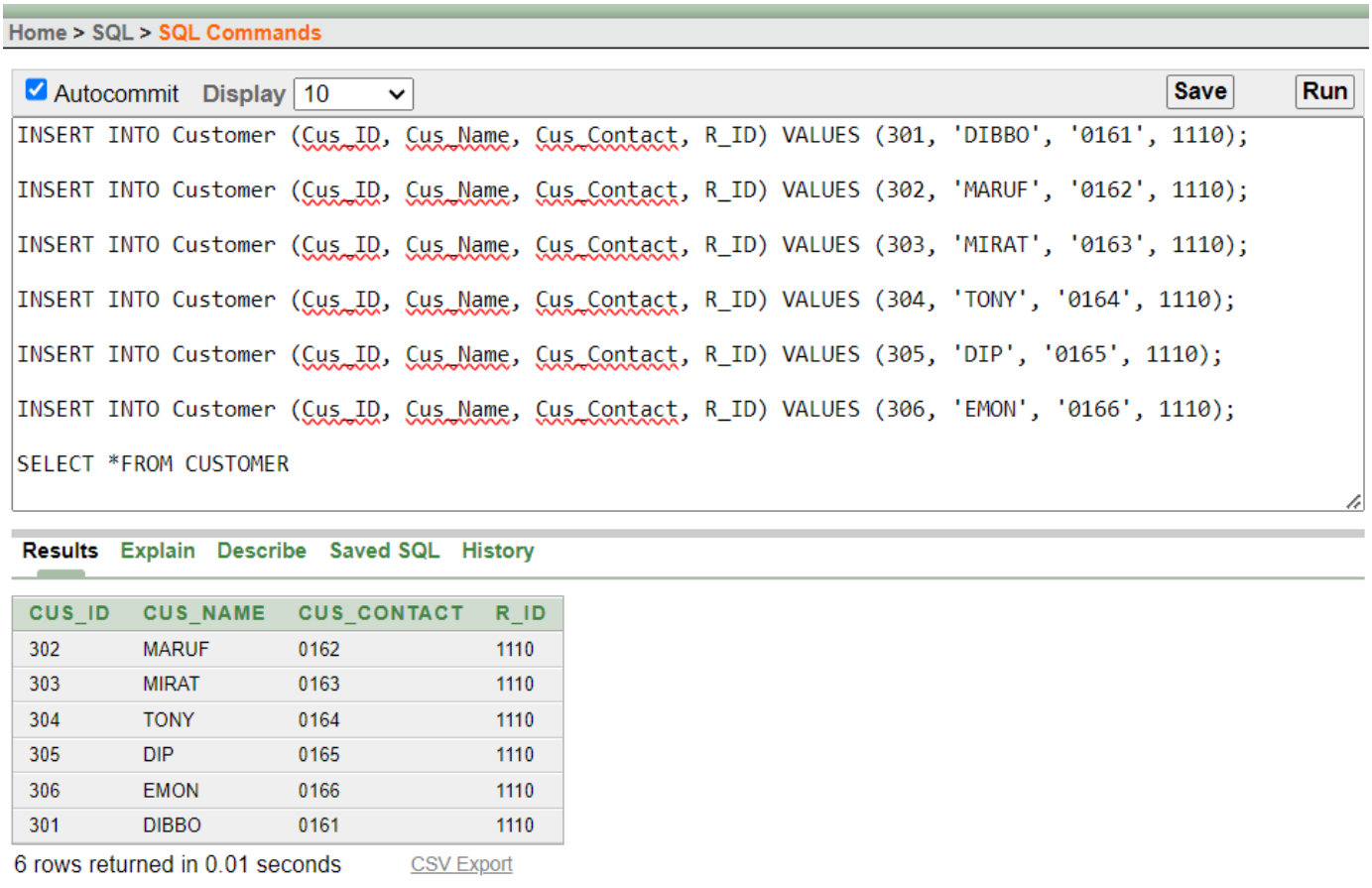
1 - 3

Data Insertion

1. Value Insertion of table Restaurant.



2. Value Insertion of table Customer.



3. Value Insertion of table Orderr.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
INSERT INTO Orderr (Order_NO, No_Items) VALUES (1, 3);

INSERT INTO Orderr (Order_NO, No_Items) VALUES (2, 9);

INSERT INTO Orderr (Order_NO, No_Items) VALUES (3, 1);

INSERT INTO Orderr (Order_NO, No_Items) VALUES (4, 4);

INSERT INTO Orderr (Order_NO, No_Items) VALUES (5, 1);

INSERT INTO Orderr (Order_NO, No_Items) VALUES (6, 1);

SELECT *FROM ORDERR
```

Results Explain Describe Saved SQL History

ORDER_NO	NO_ITEMS
1	3
2	9
3	1
4	4
5	1
6	1

6 rows returned in 0.01 seconds CSV Export

4. Value Insertion of table OrderINFO.

Autocommit Display 10

```
INSERT INTO OrderINFO (No_Items, Ord_Time) VALUES (1, '5MINS');

INSERT INTO OrderINFO (No_Items, Ord_Time) VALUES (2, '2MINS');

INSERT INTO OrderINFO (No_Items, Ord_Time) VALUES (3, '10MINS');

INSERT INTO OrderINFO (No_Items, Ord_Time) VALUES (4, '2MINS');

INSERT INTO OrderINFO (No_Items, Ord_Time) VALUES (5, '10MINS');

INSERT INTO OrderINFO (No_Items, Ord_Time) VALUES (9, '20MINS');
SELECT* FROM OrderINFO
```

Results Explain Describe Saved SQL History

NO_ITEMS	ORD_TIME
1	5MINS
2	2MINS
3	10MINS
4	2MINS
5	10MINS
9	20MINS

6 rows returned in 0.00 seconds CSV Export

5. Value Insertion of table Waiter.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
INSERT INTO Waiter (W_ID, W_Name, Cus_ID, Order_No) VALUES (1001, 'ROY', 303, 1);

INSERT INTO Waiter (W_ID, W_Name, Cus_ID, Order_No) VALUES (1002, 'RAJ', 302, 2);

INSERT INTO Waiter (W_ID, W_Name, Cus_ID, Order_No) VALUES (1003, 'NILOY', 301, 5);

INSERT INTO Waiter (W_ID, W_Name, Cus_ID, Order_No) VALUES (1004, 'KARIM', 305, 3);

INSERT INTO Waiter (W_ID, W_Name, Cus_ID, Order_No) VALUES (1005, 'SHANTO', 306, 2);

INSERT INTO Waiter (W_ID, W_Name, Cus_ID, Order_No) VALUES (1006, 'ROY', 304, 4);

INSERT INTO Waiter (W_ID, W_Name, Cus_ID, Order_No) VALUES (1007, 'JUHAB', 302, 3);

INSERT INTO Waiter (W_ID, W_Name, Cus_ID, Order_No) VALUES (1008, 'SAGAR', 303, 5);
SELECT *FROM WAITER
```

Results Explain Describe Saved SQL History

W_ID	W_NAME	CUS_ID	ORDER_NO
1001	ROY	303	1
1002	RAJ	302	2
1003	NILOY	301	5
1004	KARIM	305	3
1005	SHANTO	306	2
1006	ROY	304	4
1007	JUHAB	302	3
1008	SAGAR	303	5

8 rows returned in 0.00 seconds CSV Export

6. Value Insertion of table RestaurantINFO.

Autocommit Display 10 Save Run

```
INSERT INTO RestaurantINFO (R_Name, Address) VALUES ('CHEFS TABLE COURTSIDE', 'GULSHAN');

INSERT INTO RestaurantINFO (R_Name, Address) VALUES ('HAWA', 'PURBACHAL');

INSERT INTO RestaurantINFO (R_Name, Address) VALUES ('KHANAS', 'UTTARA');

INSERT INTO RestaurantINFO (R_Name, Address) VALUES ('CHILLOX', 'DHANMONDHI');

INSERT INTO RestaurantINFO (R_Name, Address) VALUES ('THE GREEN LOUNGE', 'SHAHBAGH');

INSERT INTO RestaurantINFO (R_Name, Address) VALUES ('ITALIAN', 'KHILKHET');

SELECT *FROM RESTAURANTINFO
```

Results Explain Describe Saved SQL History

R_NAME	ADDRESS
CHEFS TABLE COURTSIDE	GULSHAN
HAWA	PURBACHAL
KHANAS	UTTARA
CHILLOX	DHANMONDHI
THE GREEN LOUNGE	SHAHBAGH
ITALIAN	KHILKHET

6 rows returned in 0.00 seconds CSV Export

7. Value Insertion of table Food.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
INSERT INTO Food (Food_NO, Quantity, Description, Order_NO) VALUES (800, 2, 'FISH_CUTLET', 5);

INSERT INTO Food (Food_NO, Quantity, Description, Order_NO) VALUES (809, 2, 'CHICKEN_WITH_BREADBALL', 2);

INSERT INTO Food (Food_NO, Quantity, Description, Order_NO) VALUES (810, 1, 'CORN_KABAB', 3);

INSERT INTO Food (Food_NO, Quantity, Description, Order_NO) VALUES (815, 2, 'PANEER_MASALA', 4);

INSERT INTO Food (Food_NO, Quantity, Description, Order_NO) VALUES (816, 2, 'TANDOORI_MUSHROOM', 1);

INSERT INTO Food (Food_NO, Quantity, Description, Order_NO) VALUES (819, 3, 'MALAI_KOFTA', 6);

SELECT *FROM FOOD
```

Results Explain Describe Saved SQL History

FOOD_NO	QUANTITY	DESCRIPTION	ORDER_NO
800	2	FISH_CUTLET	5
809	2	CHICKEN_WITH_BREADBALL	2
810	1	CORN_KABAB	3
815	2	PANEER_MASALA	4
816	2	TANDOORI_MUSHROOM	1
819	3	MALAI_KOFTA	6

6 rows returned in 0.02 seconds CSV Export

8. Value Insertion of table Chef.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
INSERT INTO Chef (Chef_ID, Chef_Name, Order_NO) VALUES (401, 'KHAN', 1);

INSERT INTO Chef (Chef_ID, Chef_Name, Order_NO) VALUES (402, 'KHAN', 1);

INSERT INTO Chef (Chef_ID, Chef_Name, Order_NO) VALUES (403, 'JAD', 3);

INSERT INTO Chef (Chef_ID, Chef_Name, Order_NO) VALUES (404, 'JAD', 4);

INSERT INTO Chef (Chef_ID, Chef_Name, Order_NO) VALUES (405, 'PRODIP', 1);

INSERT INTO Chef (Chef_ID, Chef_Name, Order_NO) VALUES (406, 'SAM', 6);

SELECT *FROM CHEF
```

Results Explain Describe Saved SQL History

CHEF_ID	CHEF_NAME	ORDER_NO
401	KHAN	1
402	KHAN	1
403	JAD	3
404	JAD	4
405	PRODIP	1
406	SAM	6

6 rows returned in 0.00 seconds CSV Export

9. Value Insertion of table Bill.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
INSERT INTO Bill (B_NO, B_Price, Ord_Detail, Cus_ID) VALUES (10, 600, 'FISH_CUTLET', 301);

INSERT INTO Bill (B_NO, B_Price, Ord_Detail, Cus_ID) VALUES (12, 350, 'PANEER_MASALA', 301);

INSERT INTO Bill (B_NO, B_Price, Ord_Detail, Cus_ID) VALUES (14, 250, 'TANDOORI_MUSHROOM', 305);

INSERT INTO Bill (B_NO, B_Price, Ord_Detail, Cus_ID) VALUES (16, 800, 'MALAI_KOFTA', 303);

INSERT INTO Bill (B_NO, B_Price, Ord_Detail, Cus_ID) VALUES (18, 800, 'MALAI_KOFTA', 304);

INSERT INTO Bill (B_NO, B_Price, Ord_Detail, Cus_ID) VALUES (20, 600, 'FISH_CUTLET', 303);

SELECT *FROM BILL
```

Results Explain Describe Saved SQL History

B_NO	B_PRICE	ORD_DETAIL	CUS_ID
10	600	FISH_CUTLET	301
12	350	PANEER_MASALA	301
14	250	TANDOORI_MUSHROOM	305
16	800	MALAI_KOFTA	303
18	800	MALAI_KOFTA	304
20	600	FISH_CUTLET	303

6 rows returned in 0.00 seconds CSV Export

10. Value Insertion of table BillDescription.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
INSERT INTO BillDescription (Price, Vat) VALUES (250, '5');

INSERT INTO BillDescription (Price, Vat) VALUES (350, '10');

INSERT INTO BillDescription (Price, Vat) VALUES (600, '20');

INSERT INTO BillDescription (Price, Vat) VALUES (800, '20');


SELECT *FROM BILLDESCRIPTION
```

Results Explain Describe Saved SQL History

PRICE	VAT
250	5
350	10
600	20
800	20

4 rows returned in 0.00 seconds CSV Export

11. Value Insertion of table FoodDetails.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
INSERT INTO FoodDetails (Food_NO, Quantity, Price) VALUES (800, 2, 250);

INSERT INTO FoodDetails (Food_NO, Quantity, Price) VALUES (800, 3, 350);

INSERT INTO FoodDetails (Food_NO, Quantity, Price) VALUES (810, 1, 800);

INSERT INTO FoodDetails (Food_NO, Quantity, Price) VALUES (810, 4, 600);

INSERT INTO FoodDetails (Food_NO, Quantity, Price) VALUES (815, 5, 600);

INSERT INTO FoodDetails (Food_NO, Quantity, Price) VALUES (816, 6, 800);

SELECT *FROM FOODDETAILS
```

Results Explain Describe Saved SQL History

QUANTITY	PRICE	FOOD_NO
2	250	800
3	350	800
1	800	810
4	600	810
5	600	815
6	800	816

6 rows returned in 0.00 seconds CSV Export

12. Value Insertion of table ManagerDetails.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
INSERT INTO ManagerDetails (R_ID, Man_ID) VALUES (1110, 2248);

INSERT INTO ManagerDetails (R_ID, Man_ID) VALUES (1111, 2249);

INSERT INTO ManagerDetails (R_ID, Man_ID) VALUES (1112, 2248);

INSERT INTO ManagerDetails (R_ID, Man_ID) VALUES (1113, 2251);

INSERT INTO ManagerDetails (R_ID, Man_ID) VALUES (1114, 2252);

INSERT INTO ManagerDetails (R_ID, Man_ID) VALUES (1115, 2251);

SELECT *FROM MANAGERDETAILS
```

Results Explain Describe Saved SQL History

R_ID	MAN_ID
1110	2248
1111	2249
1112	2248
1113	2251
1114	2252
1115	2251

6 rows returned in 0.01 seconds CSV Export

13. Value Insertion of table WaiterManage.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
INSERT INTO WaiterManage (W_ID, W_Name, Man_ID) VALUES (1001, 'ROY', 2248);

INSERT INTO WaiterManage (W_ID, W_Name, Man_ID) VALUES (1002, 'RAJ', 2249);

INSERT INTO WaiterManage (W_ID, W_Name, Man_ID) VALUES (1003, 'NILOY', 2251);

INSERT INTO WaiterManage (W_ID, W_Name, Man_ID) VALUES (1004, 'KARIM', 2252);

INSERT INTO WaiterManage (W_ID, W_Name, Man_ID) VALUES (1005, 'SHANTO', 2250);

INSERT INTO WaiterManage (W_ID, W_Name, Man_ID) VALUES (1006, 'ROY', 2253);

SELECT *FROM WAITERMANAGE
```

Results Explain Describe Saved SQL History

W_ID	W_NAME	MAN_ID
1001	ROY	2248
1002	RAJ	2249
1003	NILOY	2251
1004	KARIM	2252
1005	SHANTO	2250
1006	ROY	2253

6 rows returned in 0.02 seconds CSV Export

14. Value Insertion of table Manager.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
INSERT INTO Manager (Man_ID, Man_Name) VALUES (2248, 'RONY');

INSERT INTO Manager (Man_ID, Man_Name) VALUES (2249, 'JON');

INSERT INTO Manager (Man_ID, Man_Name) VALUES (2250, 'RAKIB');

INSERT INTO Manager (Man_ID, Man_Name) VALUES (2251, 'ESHAN');

INSERT INTO Manager (Man_ID, Man_Name) VALUES (2252, 'RONY');

INSERT INTO Manager (Man_ID, Man_Name) VALUES (2253, 'NIROB');
SELECT *FROM MANAGER
```

Results Explain Describe Saved SQL History

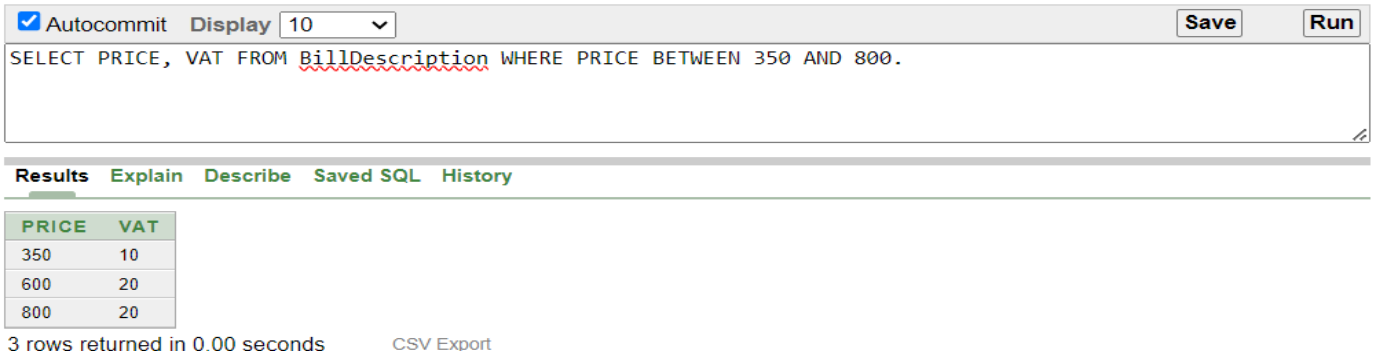
MAN_ID	MAN_NAME
2248	RONY
2249	JON
2250	RAKIB
2251	ESHAN
2252	RONY
2253	NIROB

6 rows returned in 0.00 seconds CSV Export

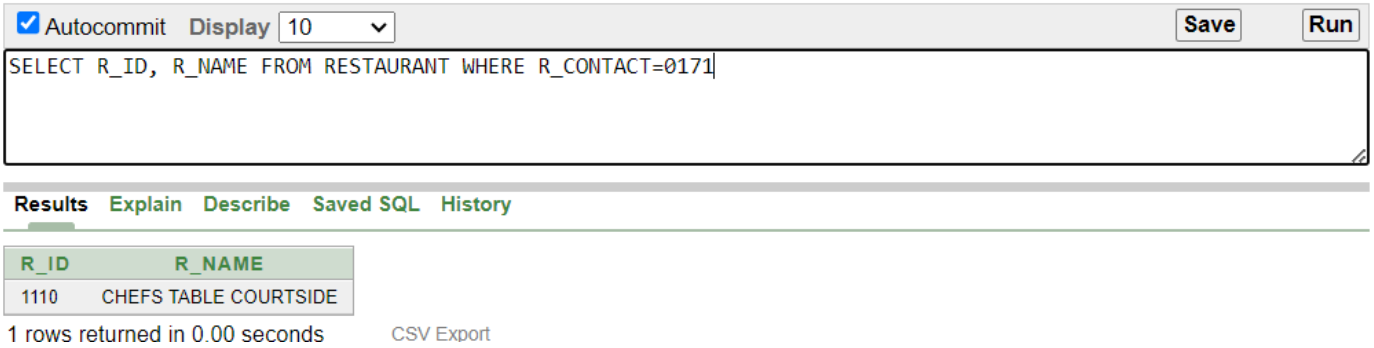
QUERY TEST

Simple Query:

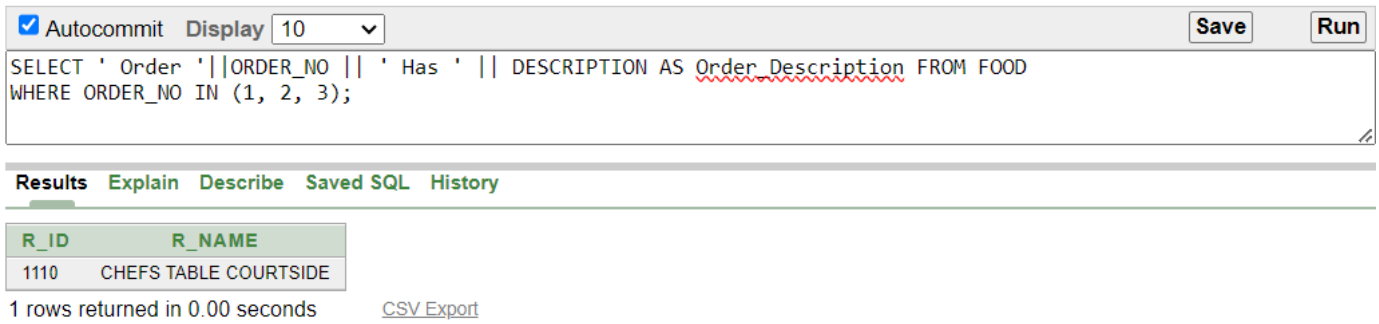
1. **Question:** Retrieve the price and VAT information from the BillDescription table for items with a price between $350 and $800.



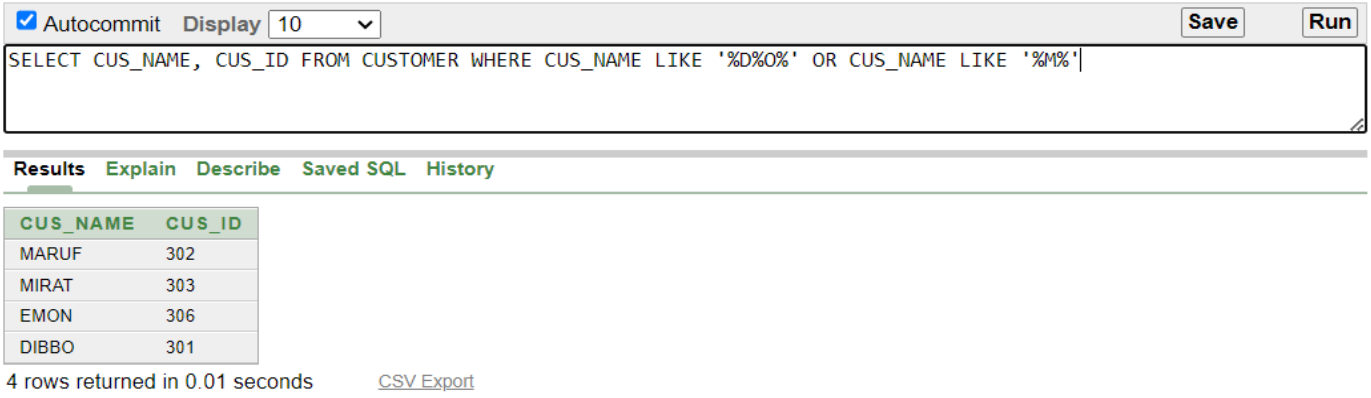
2. **Question:** Retrieve the Restaurant ID and Restaurant Name for the restaurant with the contact number '0171'.



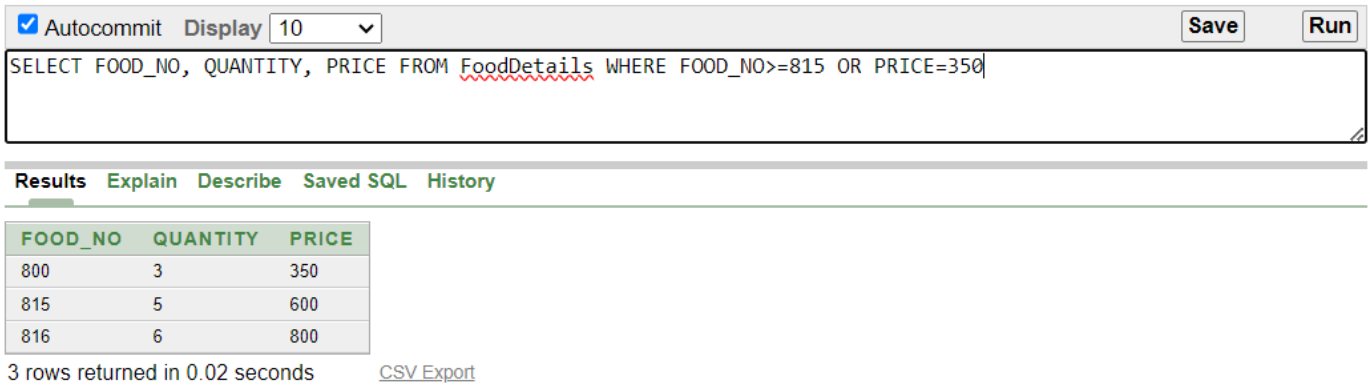
3. **Question:** Display concatenated description for each food item in orders with Order Numbers 1, 2, and 3? The description should include the following format: ' Order [Order_Number] Has [Description]'.



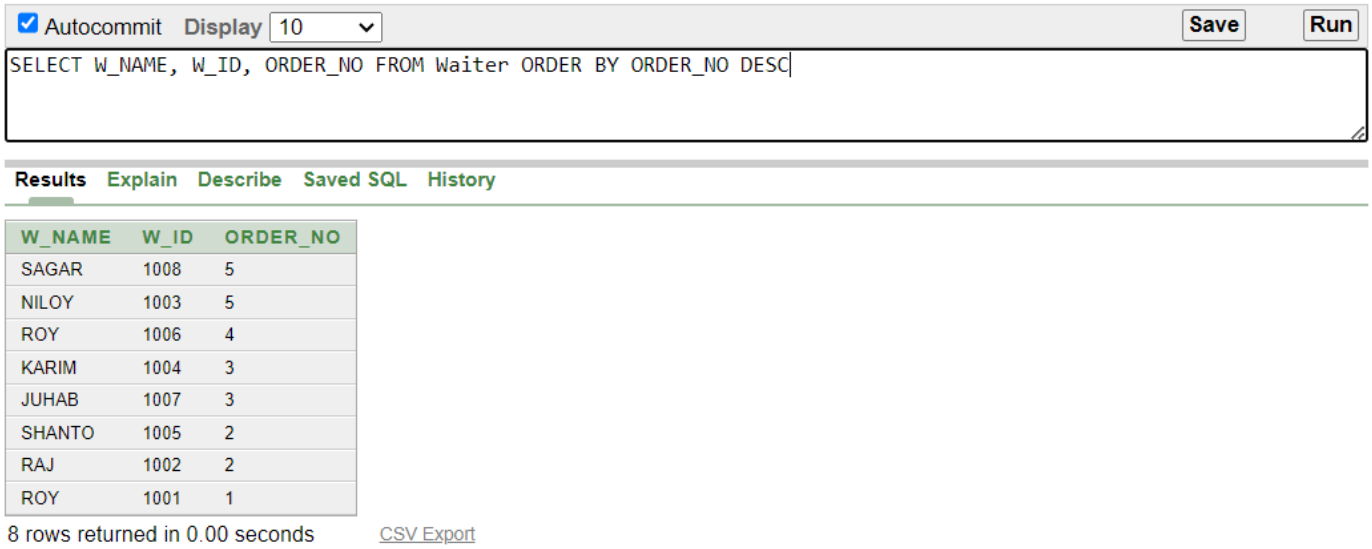
4. **Question:** List the Customer Names and IDs for customers whose names contain 'D' and 'O' together or have a name starting with any character followed by 'M'.



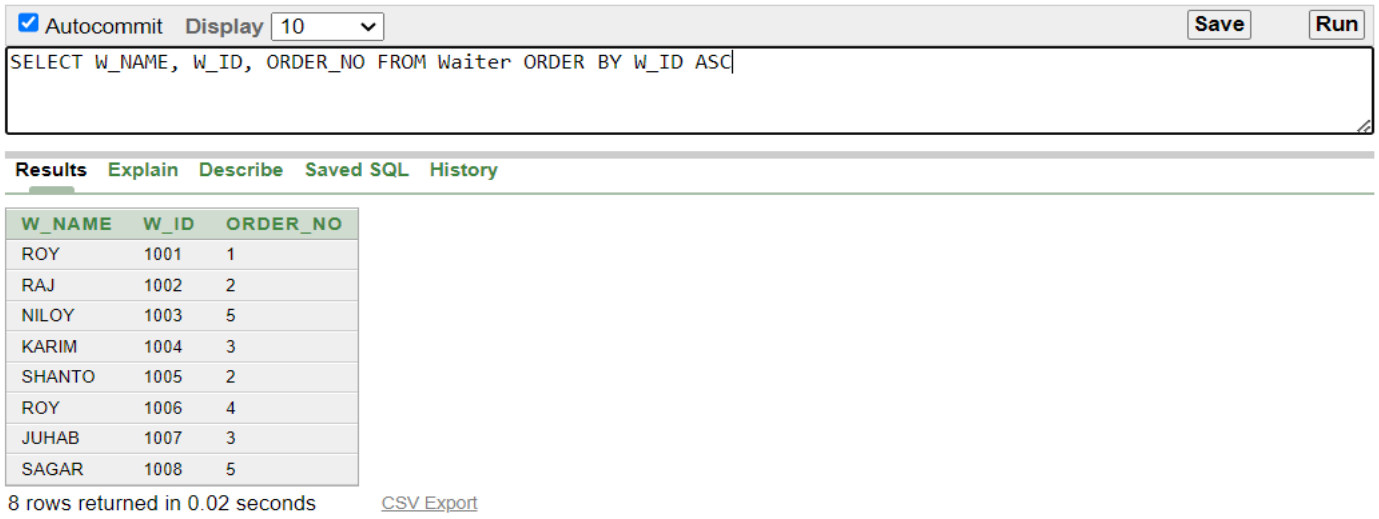
5. **Question:** List the Food Numbers, Quantities, and Prices for items where the Food Number is 815 or greater, or the Price is 350



6. **Question:** Display Waiter Name, Waiter ID and Order Number sorted by descending order of their associated order numbers?



7. **Question:** Retrieve Waiter Name, Waiter ID, and Order Number, order by ascending Order Numbers and then in descending Waiter IDs within each order.



Autocommit Display 10 Save Run

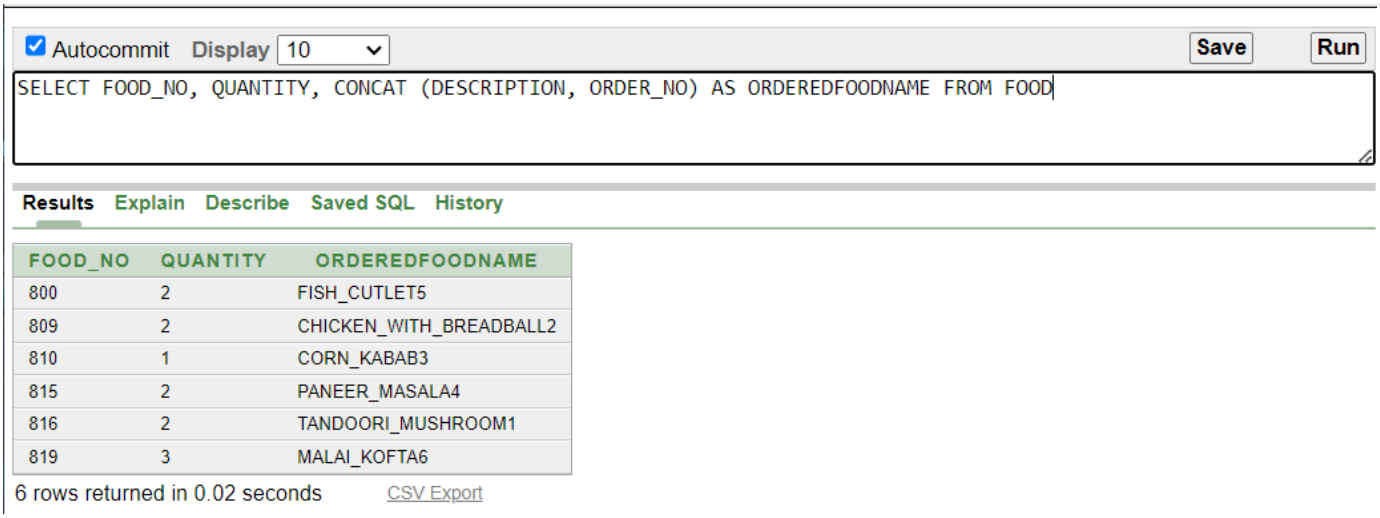
```
SELECT W_NAME, W_ID, ORDER_NO FROM Waiter ORDER BY W_ID ASC
```

Results Explain Describe Saved SQL History

W_NAME	W_ID	ORDER_NO
ROY	1001	1
RAJ	1002	2
NILOY	1003	5
KARIM	1004	3
SHANTO	1005	2
ROY	1006	4
JUHAB	1007	3
SAGAR	1008	5

8 rows returned in 0.02 seconds CSV Export

8. **Question:** Display the Food Number, Quantity, and a concatenated string of Description and Order Number as (ORDEREDFOODNAME) for each record in the FOOD table?



Autocommit Display 10 Save Run

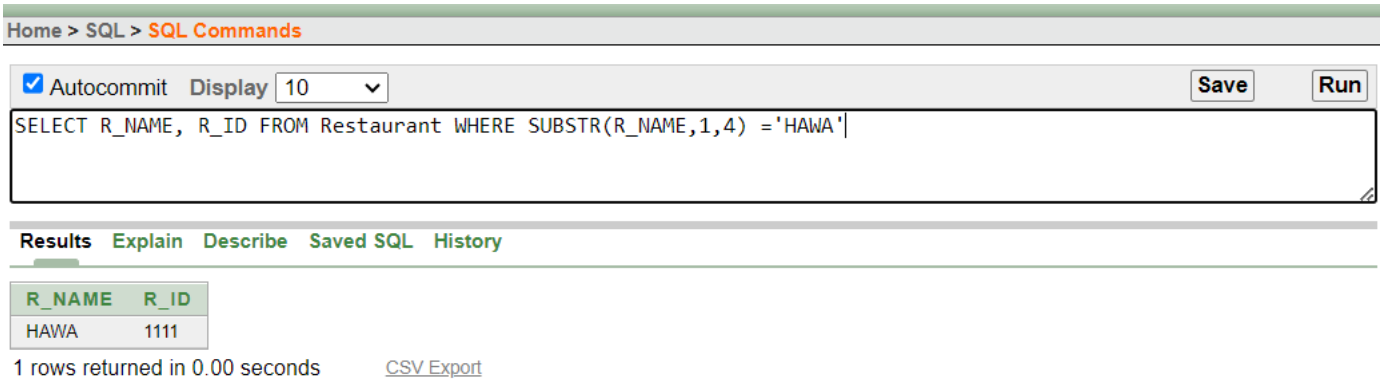
```
SELECT FOOD_NO, QUANTITY, CONCAT (DESCRIPTION, ORDER_NO) AS ORDEREDFOODNAME FROM FOOD
```

Results Explain Describe Saved SQL History

FOOD_NO	QUANTITY	ORDEREDFOODNAME
800	2	FISH_CUTLET5
809	2	CHICKEN_WITH_BREADBALL2
810	1	CORN_KABAB3
815	2	PANEER_MASALA4
816	2	TANDOORI_MUSHROOM1
819	3	MALAI_KOFTA6

6 rows returned in 0.02 seconds CSV Export

9. **Question:** Retrieve the Restaurant Name and Restaurant ID for restaurants whose names start with 'HAWA'.



Home > SQL > SQL Commands

Autocommit Display 10 Save Run

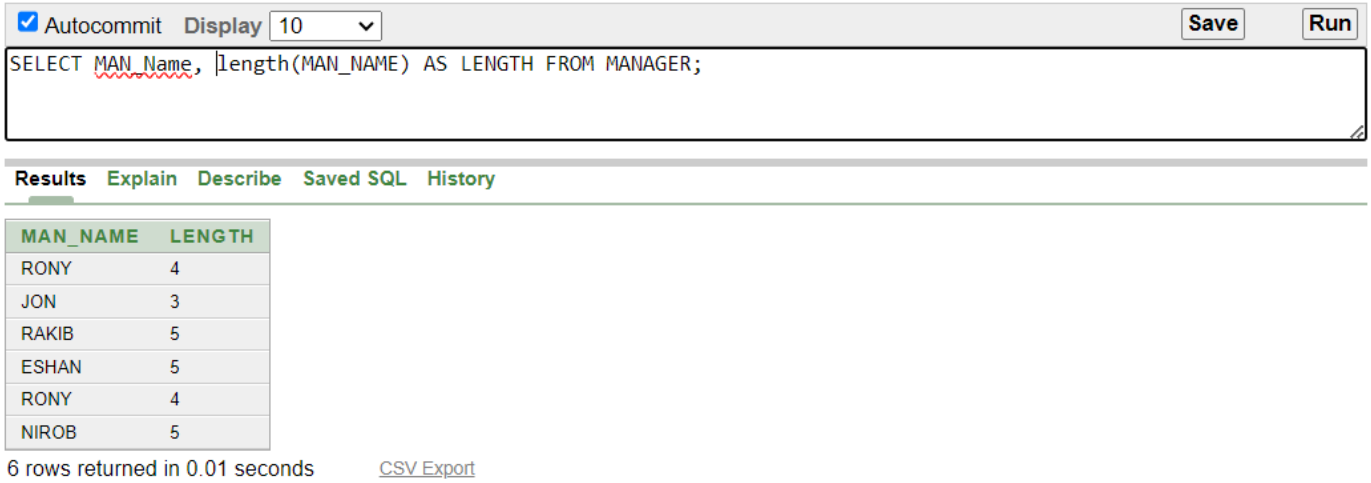
```
SELECT R_NAME, R_ID FROM Restaurant WHERE SUBSTR(R_NAME,1,4) ='HAWA'
```

Results Explain Describe Saved SQL History

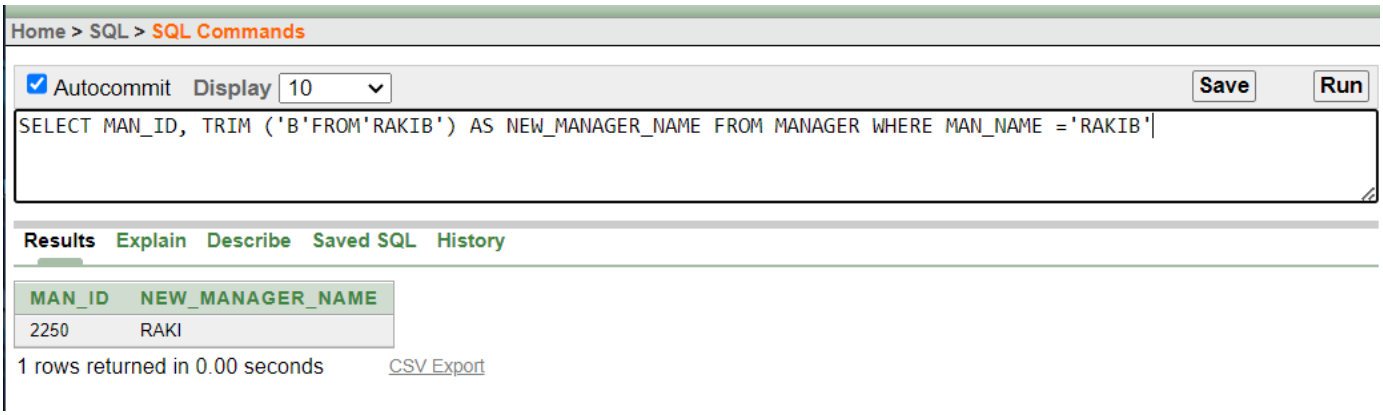
R_NAME	R_ID
HAWA	1111

1 rows returned in 0.00 seconds CSV Export

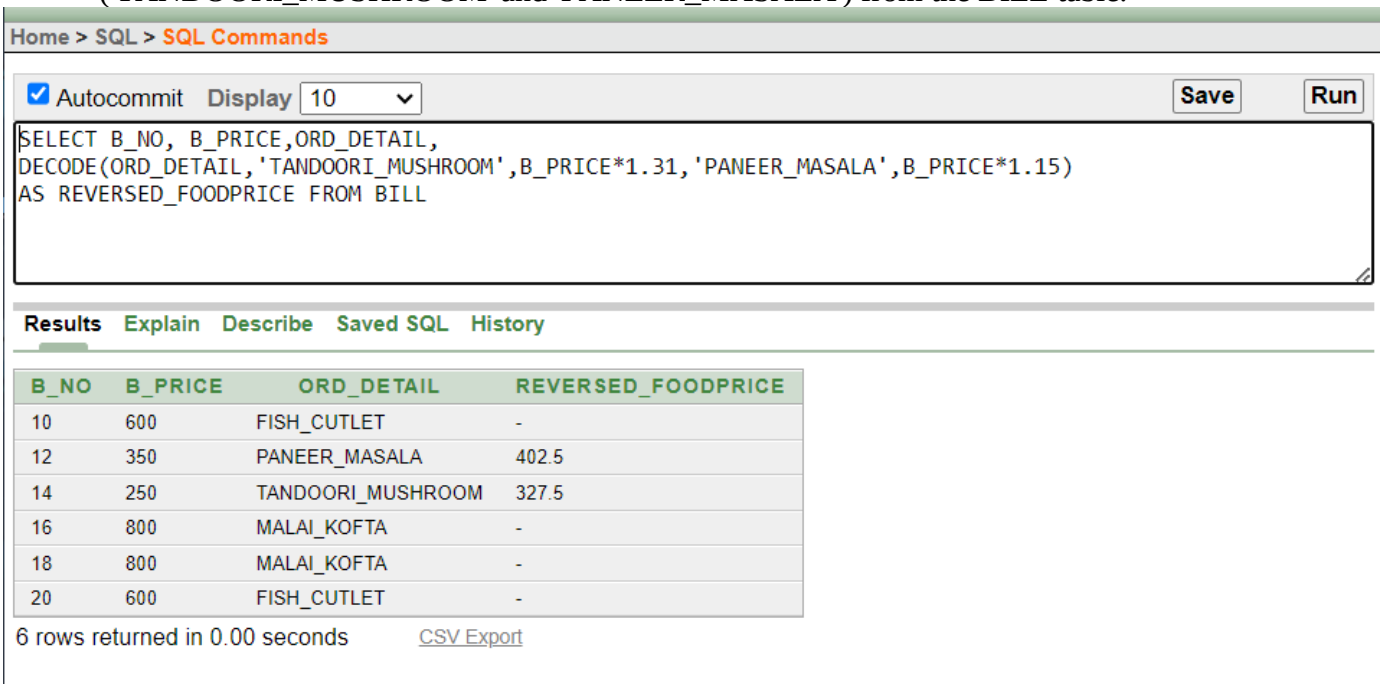
10. **Question:** Provide the Manager Name (MAN_Name) and the length of each manager's name (LENGTH) from the MANAGER table.



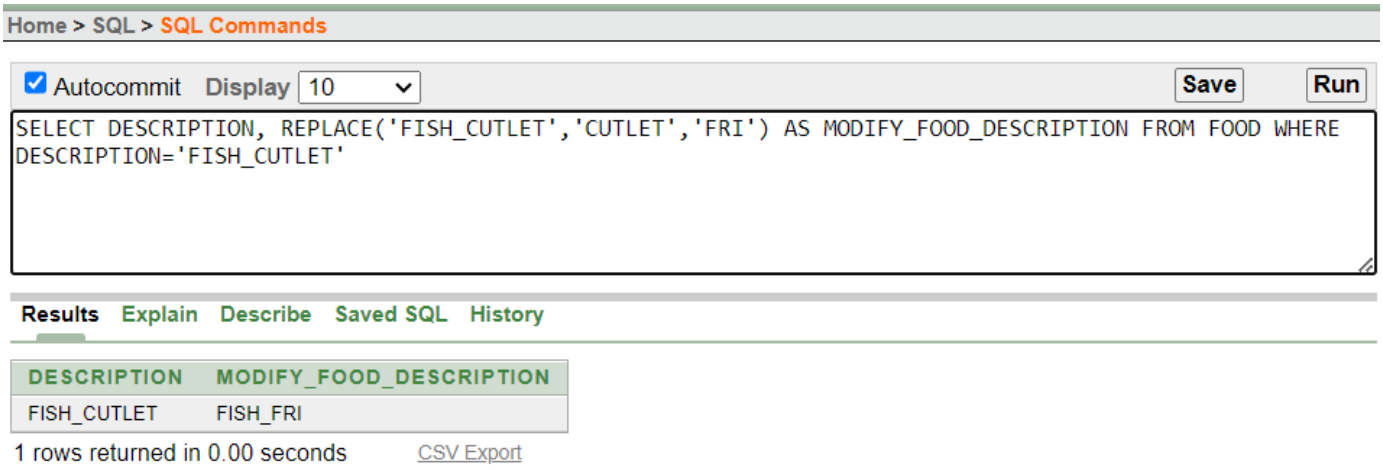
11. **Question:** Retrieve the Manager ID and a modified version of the Manager Name (NEW_MANAGER_NAME) removing the letter 'B' from the name, for managers with the original name 'RAKIB.



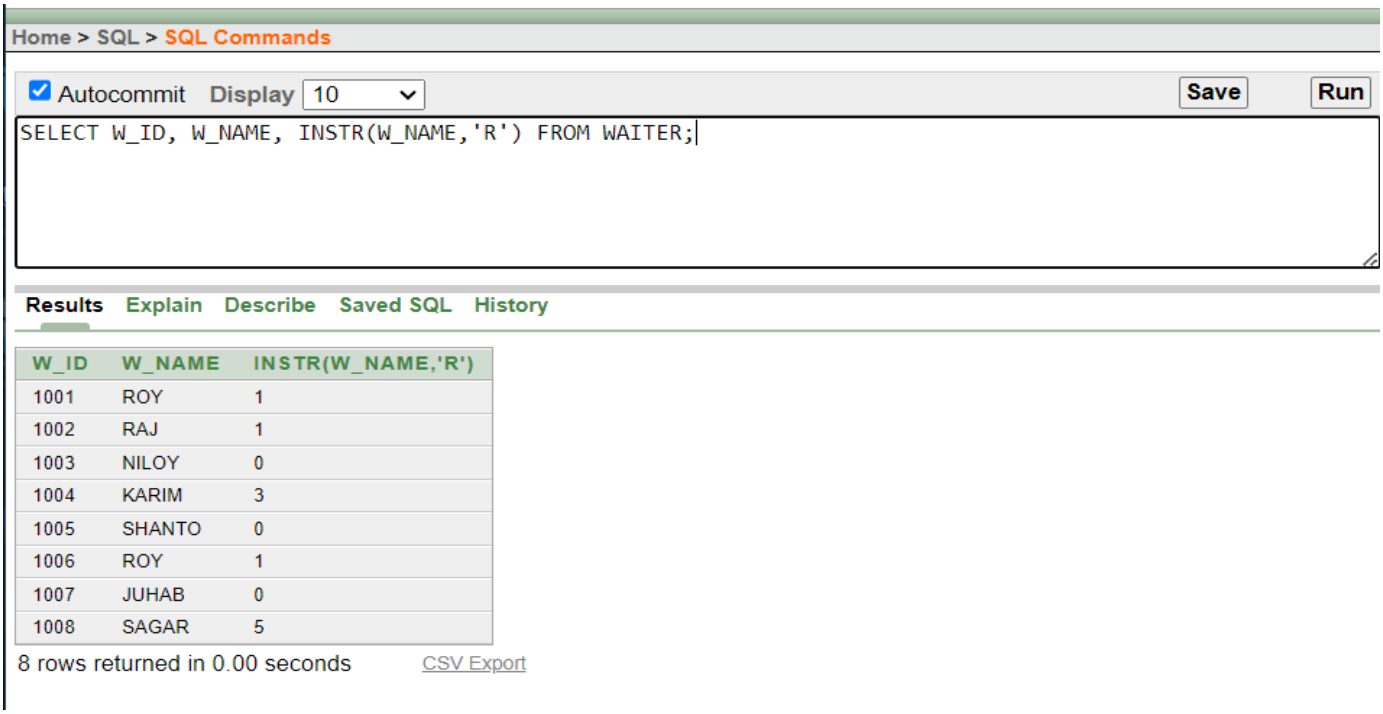
12. **Question:** Fetch the Bill Number, Bill Price, Order Detail, and a reversed food price calculation using the DECODE function, considering different multipliers based on specific order details ('TANDOORI_MUSHROOM' and 'PANEER_MASALA') from the BILL table.



13. **Question:** Retrieve the original food Description and a modified version (MODIFY_FOOD_DESCRIPTION) using the REPLACE function to replace 'CUTLET' with 'FRI' for items with the original description 'FISH_CUTLET.'

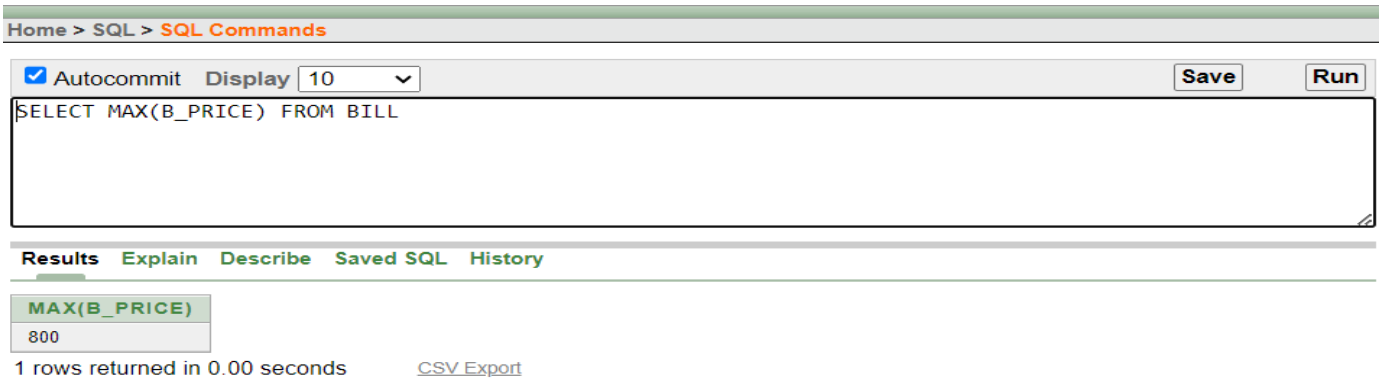


14. **Question:** Name, and the position (index) of the letter 'R' in the Waiter Name from the WAITER table.

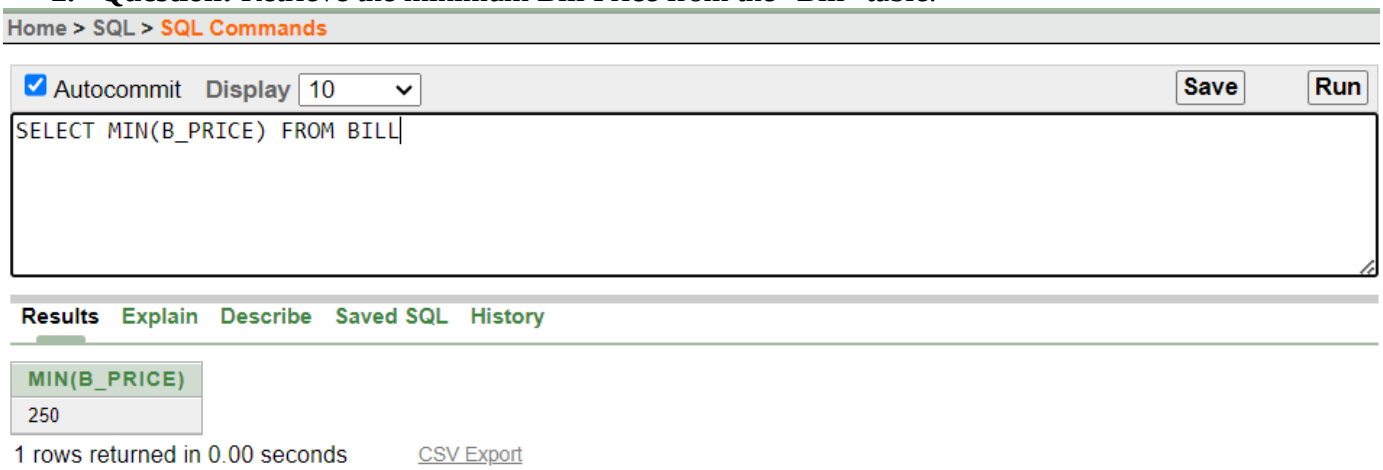


Aggregate Query

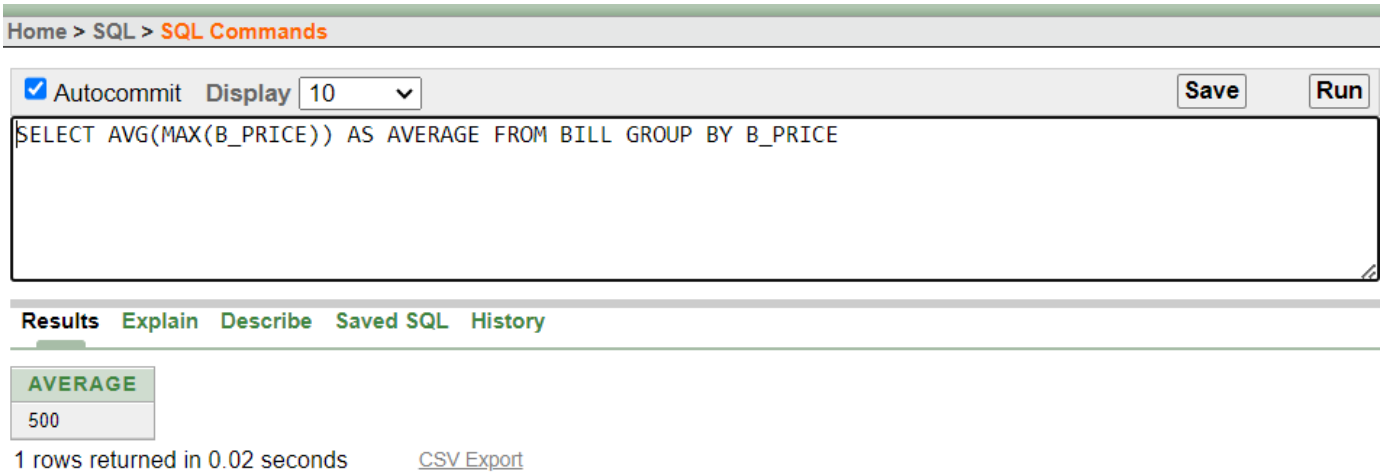
1. **Question:** Retrieve the maximum Bill Price from the "Bill" table.



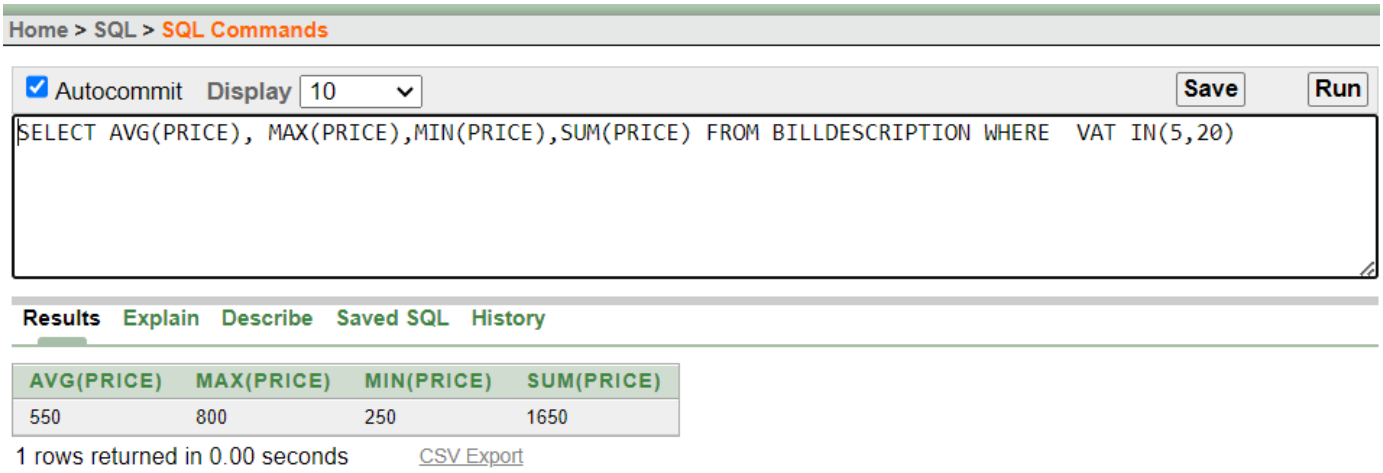
2. **Question:** Retrieve the minimum Bill Price from the "Bill" table.



3. **Question:** Calculate the average of the maximum Bill Price for each distinct Bill Price group in the "Bill" table.

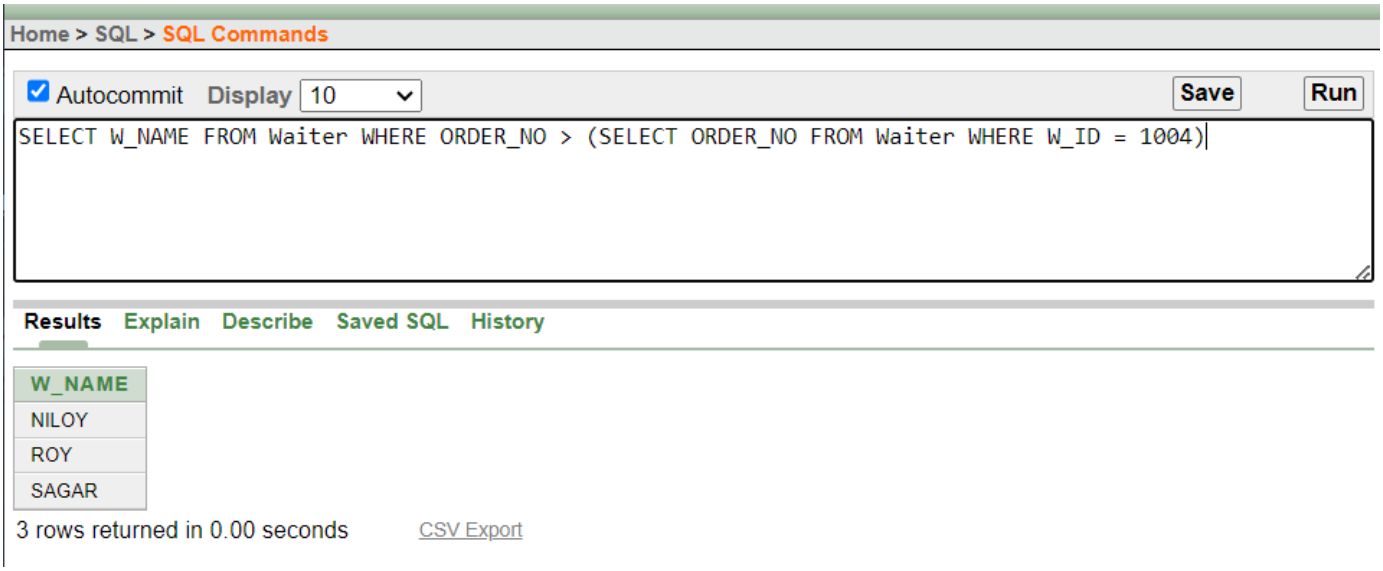


4. **Question:** Calculate the average, maximum, minimum, and total (sum) Price for items in the "BillDescription" table where the VAT is either 5 or 20.

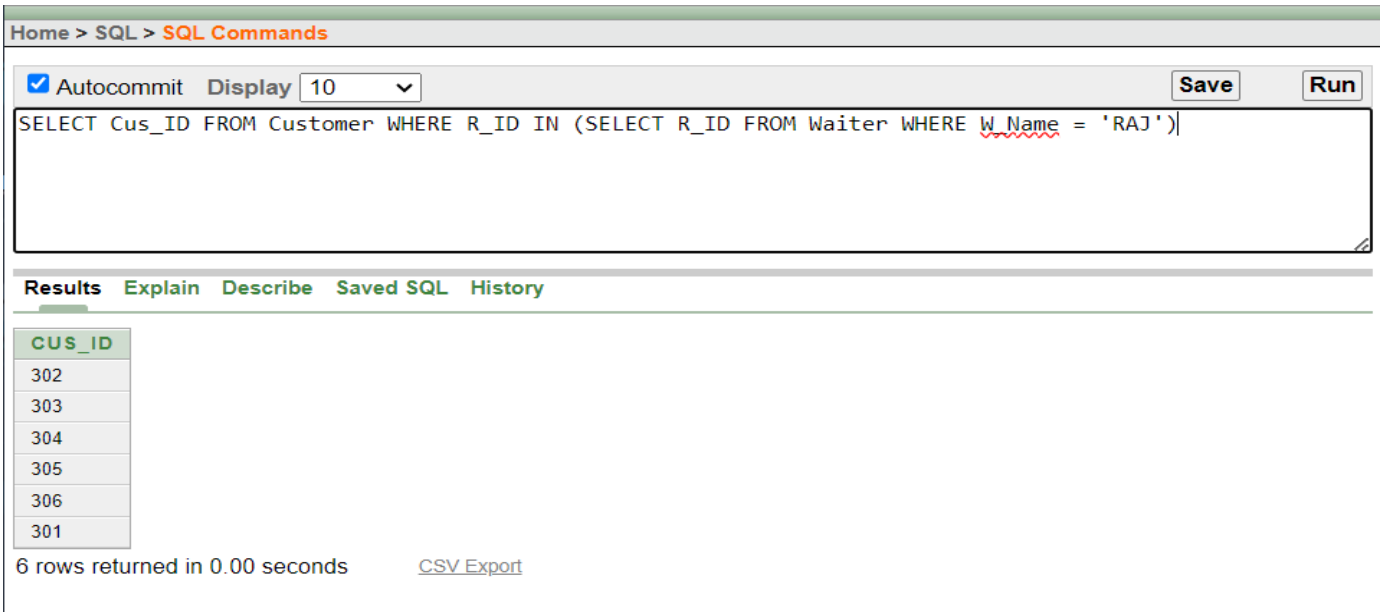


Subquery (Multiple-row subquery and Single -row subquery)

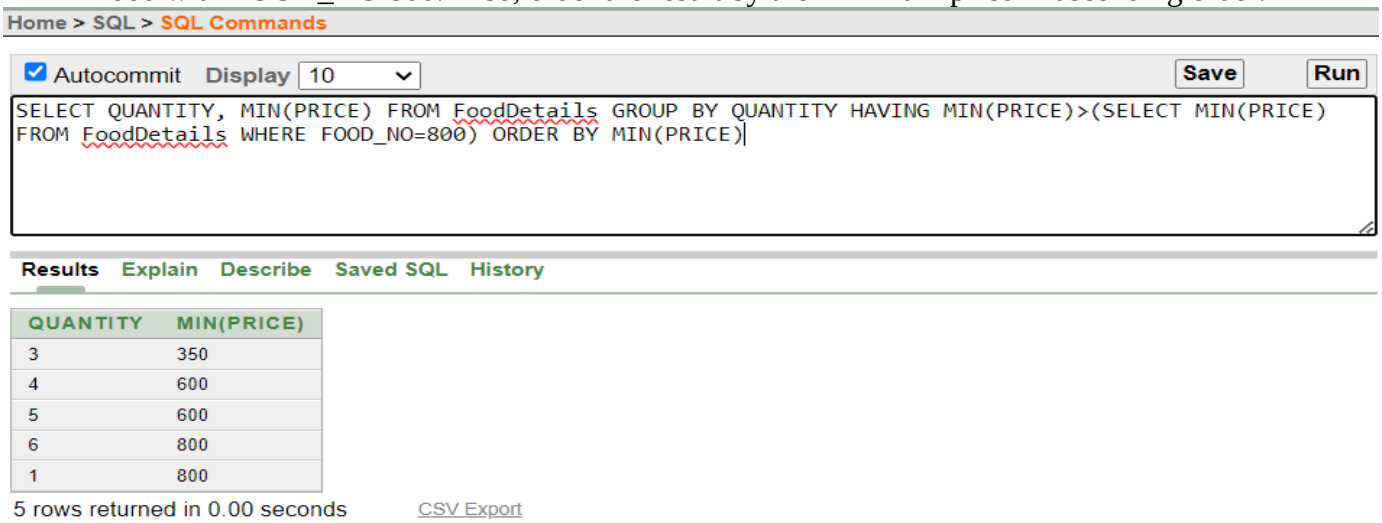
1. **Question:** Display the names of waiters who handled orders placed after the order was handled by Waiter ID 1004.



2. **Question:** Retrieve the Customer ID of those who placed orders with Waiter Name 'RAJ'.



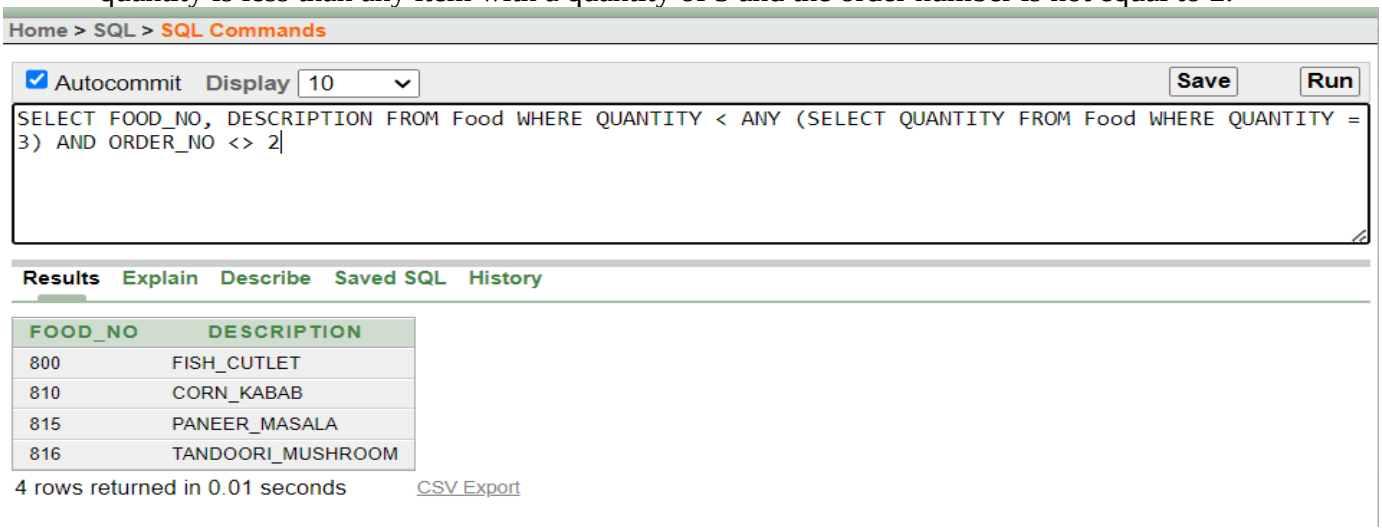
3. **Question:** Retrieve the quantity and the minimum price for each quantity from the "FoodDetails" table, considering only those rows where the minimum price is greater than the minimum price for food with FOOD_NO 800. Also, order the result by the minimum price in ascending order.



QUANTITY	MIN(PRICE)
3	350
4	600
5	600
6	800
1	800

5 rows returned in 0.00 seconds CSV Export

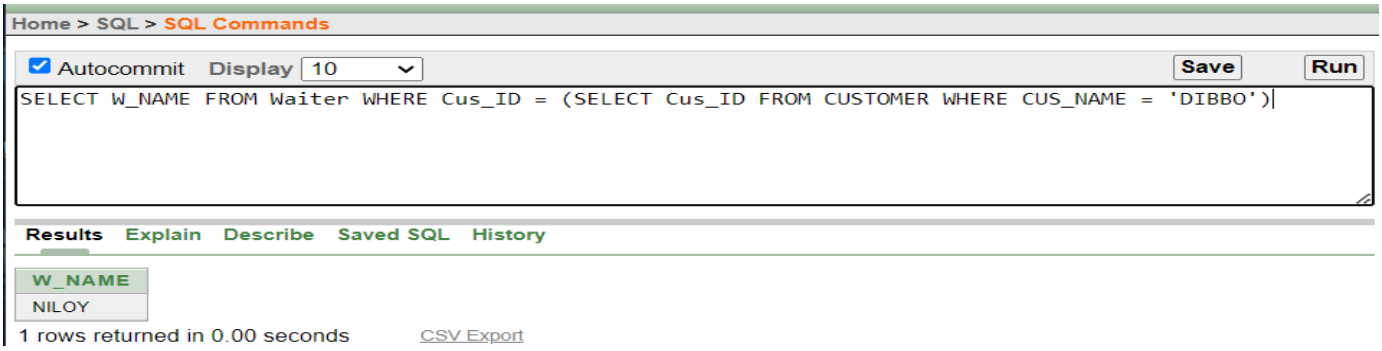
4. **Question:** Retrieve the Food Number and Description from the "Food" table for items where the quantity is less than any item with a quantity of 3 and the order number is not equal to 2.



FOOD_NO	DESCRIPTION
800	FISH_CUTLET
810	CORN_KABAB
815	PANEER_MASALA
816	TANDOORI_MUSHROOM

4 rows returned in 0.01 seconds CSV Export

5. **Question:** Retrieve the Waiter Name (W_NAME) from the "Waiter" table for the waiter associated with the customer whose name is 'DIBBO'.

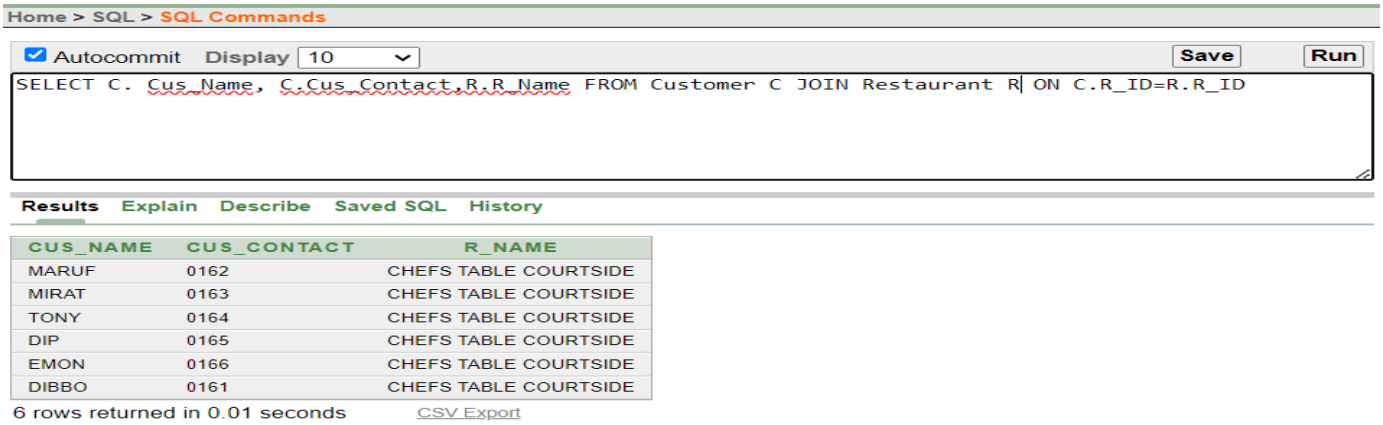


W_NAME
NILOY

1 rows returned in 0.00 seconds CSV Export

Joining

1. **Question:** Retrieve the Customer Name, Customer Contact, and Restaurant Name by joining the "Customer" table (C) with the "Restaurant" table (R) using the common column R_ID.



CUS_NAME	CUS_CONTACT	R_NAME
MARUF	0162	CHEFS TABLE COURTSIDE
MIRAT	0163	CHEFS TABLE COURTSIDE
TONY	0164	CHEFS TABLE COURTSIDE
DIP	0165	CHEFS TABLE COURTSIDE
EMON	0166	CHEFS TABLE COURTSIDE
DIBBO	0161	CHEFS TABLE COURTSIDE

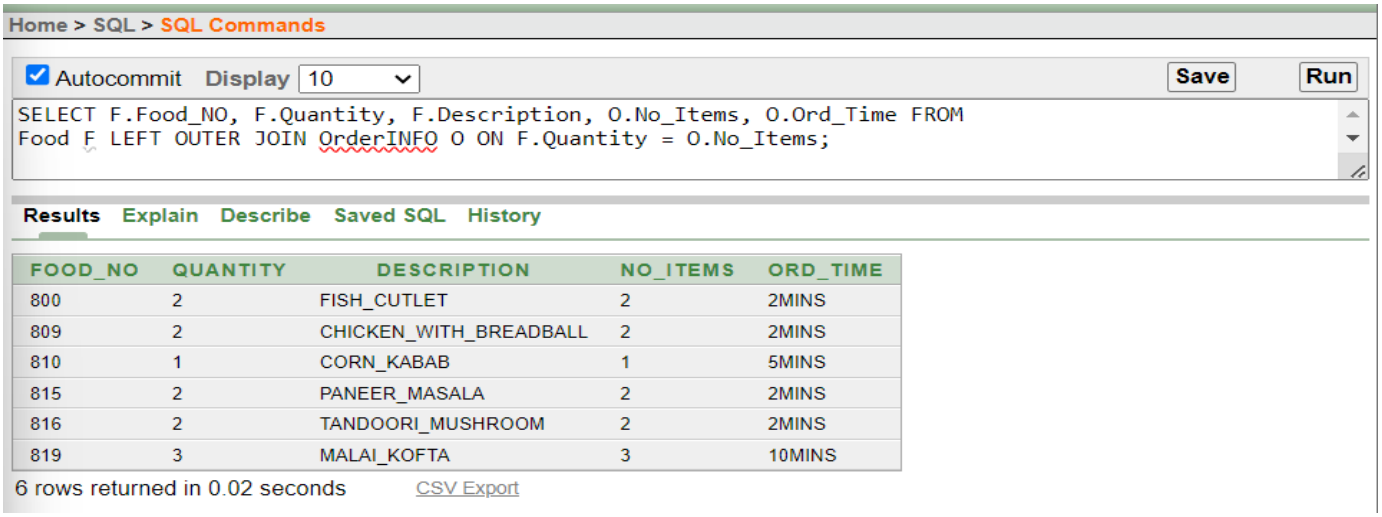
6 rows returned in 0.01 seconds CSV Export

2. **Question:** Retrieve the Waiter ID and Waiter Name by joining the "Waiter" table (W) with the "Food" table (F) using the common column ORDER_NO.

W_ID	W_NAME
1001	ROY
1002	RAJ
1003	NILOY
1004	KARIM
1005	SHANTO
1006	ROY
1007	JUHAB
1008	SAGAR

8 rows returned in 0.00 seconds CSV Export

3. **Question:** Retrieve the details of food items, including their (Food_NO), (Quantity), (Description), (No_Items), and (Ord_Time). Include all food items, even those that do not have corresponding entries in the OrderINFO table.



FOOD_NO	QUANTITY	DESCRIPTION	NO_ITEMS	ORD_TIME
800	2	FISH_CUTLET	2	2MINS
809	2	CHICKEN_WITH_BREADBALL	2	2MINS
810	1	CORN_KABAB	1	5MINS
815	2	PANEER_MASALA	2	2MINS
816	2	TANDOORI_MUSHROOM	2	2MINS
819	3	MALAI_KOFTA	3	10MINS

6 rows returned in 0.02 seconds CSV Export

4. **Question:** Provide pairs of waiters and their corresponding customer IDs who have served the same customer? The list should include the IDs and names of both waiters, along with the shared customer ID. If a waiter hasn't served alongside another waiter for the same customer, that pair won't be included in the result.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
SELECT
    w1.W_ID AS Waiter1_ID,
    w1.W_Name AS Waiter1_Name,
    w2.W_ID AS Waiter2_ID,
    w2.W_Name AS Waiter2_Name,
    w1.Cus_ID AS Customer_ID
FROM
    Waiter w1
JOIN
    Waiter w2 ON w1.Cus_ID = w2.Cus_ID
```

Results Explain Describe Saved SQL History

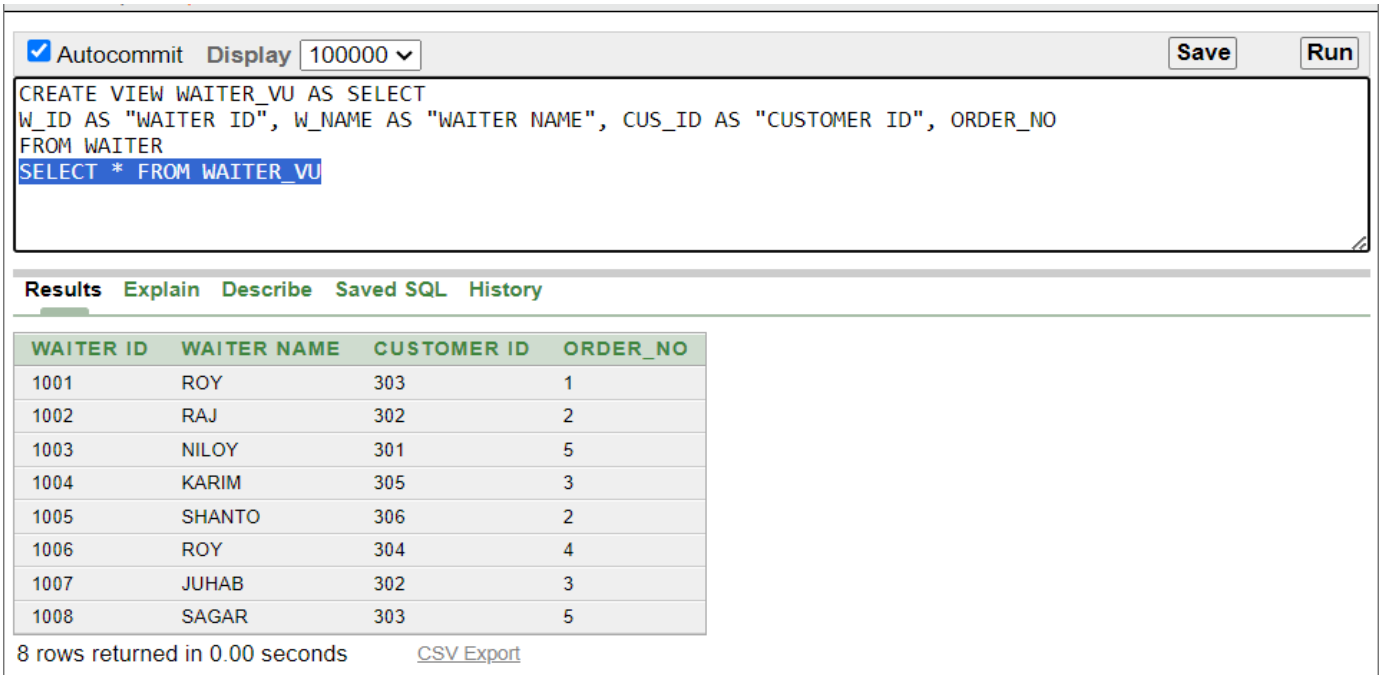
WAITER1_ID	WAITER1_NAME	WAITER2_ID	WAITER2_NAME	CUSTOMER_ID
1008	SAGAR	1001	ROY	303
1001	ROY	1001	ROY	303
1007	JUHAB	1002	RAJ	302
1002	RAJ	1002	RAJ	302
1003	NILOY	1003	NILOY	301
1004	KARIM	1004	KARIM	305
1005	SHANTO	1005	SHANTO	306
1006	ROY	1006	ROY	304
1007	JUHAB	1007	JUHAB	302
1002	RAJ	1007	JUHAB	302

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds CSV Export

SIMPLE VIEW

1. **Question:** Create a view that presents essential information about waiters, including their IDs, names, associated customer IDs, and order numbers.



COMPLEX VIEW

2. **Question:** Provide a view that shows the waiter ID, waiter name, manager name, and manager ID for each waiter who is managed by a specific manager.

Autocommit Display 100000 Save Run

```
CREATE VIEW WAITER_VU AS SELECT
W_ID AS "WAITER ID", W_NAME AS "WAITER NAME", CUS_ID AS "CUSTOMER ID", ORDER_NO
FROM WAITER
SELECT * FROM WAITER_VU
```

Results Explain Describe Saved SQL History

WAITER ID	WAITER NAME	CUSTOMER ID	ORDER_NO
1001	ROY	303	1
1002	RAJ	302	2
1003	NILOY	301	5
1004	KARIM	305	3
1005	SHANTO	306	2
1006	ROY	304	4
1007	JUHAB	302	3
1008	SAGAR	303	5

8 rows returned in 0.00 seconds CSV Export

Database connection

Setting up java-mysql connection

Step 1: Download and Install XAMPP

Download XAMPP from the official website: XAMPP Download
Install XAMPP on your machine. During installation, make sure to start the MySQL server.

Step 2: Start XAMPP MySQL Server

Open the XAMPP Control Panel.
Start the MySQL server by clicking on the "*Start*" button next to MySQL.

Step 3: Create a Database and User in MySQL

Open a web browser and go to http://localhost/phpmyadmin.
Log in to phpMyAdmin.
Create a new database. Click on "*Databases*" and enter a name for your database.
Create a new user. Click on "*User Accounts*" then "*Add user*" Enter a username and password and grant the user all privileges on the database you created.

Step 4: Download MySQL Connector/J

Download the MySQL Connector/J (JDBC driver) from the official website: MySQL Connector/J
Add the downloaded JAR file to your project in NetBeans.

Step 5: Write Java Code to Connect to MySQL

Here's a simple example of a Java program connecting to MySQL using JDBC:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class MySQLConnectionTest {
   public static void main (String [] args) {
      String jdbcUrl = "jdbc: mysql://localhost:3306/your_database";
      String username = "your_username";
      String password = "your_password";

      try (Connection connection = DriverManager.getConnection(jdbcUrl, username, password)) {
         if (connection! = null) {
            System.out.println("Connected to the database!");
         }
      } catch (SQLException e) {
```

examples:

Create table:

```java
public static void main(String[] args) {
    String url = "jdbc:mysql://127.0.0.1/youtube";
    String username = "root";
    String password = "";

    try (Connection connection = DriverManager.getConnection(url, username, password)) {
        // Create a Statement object
        Statement statement = connection.createStatement();

        // SQL query to create a table named 'Restaurant'
        String createTableQuery = "CREATE TABLE Restaurant ("
                + "R_ID INT PRIMARY KEY,"
                + "R_Name VARCHAR(100),"
                + "R_Contact VARCHAR(20))";

        statement.executeUpdate(createTableQuery);
        System.out.println("Table 'Restaurant' created successfully!");
```

test2table > main > try

Java Call Hierarchy | Output - jdbc (run)

```
run:
Table 'Restaurant' created successfully!
BUILD SUCCESSFUL (total time: 2 seconds)
```

Fig1: NetBeans

Your SQL query has been executed successfully.

```
desc Restaurant;
```

[Edit inline] [Edit] [Create PHP code]

Extra options

Field	Type	Null	Key	Default	Extra
R_ID	int(11)	NO	PRI	*NULL*	
R_Name	varchar(100)	YES		*NULL*	
R_Contact	varchar(20)	YES		*NULL*	

Query results operations

Fig2: phpMyAdmin

Data Insertion:

```
4   import java.sql.Statement;
5
6   public class test2table {
7       public static void main(String[] args) {
8           String url = "jdbc:mysql://127.0.0.1/youtube";
9           String username = "root";
10          String password = "";
11
12          try (Connection connection = DriverManager.getConnection(url, username, password)) {
13              // Create a Statement object
14              Statement statement = connection.createStatement();
15
16              // SQL query to insert values into the 'Restaurant' table
                String insertTableQuery = "INSERT INTO Restaurant (R_ID, R_Name, R_Contact) "
                        + "VALUES (1110, 'CHEFS TABLE COURTSIDE', '0171')";
19
20              // Execute the query to insert values
```

test2table > main > try > insertTableQuery

Java Call Hierarchy | Output - jdbc (run)

```
run:
Values inserted into 'Restaurant' successfully!
BUILD SUCCESSFUL (total time: 1 second)
```

Fig3: NetBeans

Showing rows 0 - 5 (6 total, Query took 0.0271 seconds.)

```
SELECT* from Restaurant;
```

[Edit inline] [Edit] [Create PHP code]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	R_ID	R_Name	R_Contact
Edit Copy Delete	1110	CHEFS TABLE COURTSIDE	0171
Edit Copy Delete	1111	HAWA	0172
Edit Copy Delete	1112	KHANAS	0173
Edit Copy Delete	1113	CHILLOX	0174
Edit Copy Delete	1114	THE GREEN LOUNGE	0175
Edit Copy Delete	1115	ITALIAN	0176

Fig4: phpMyAdmin

Show full table:

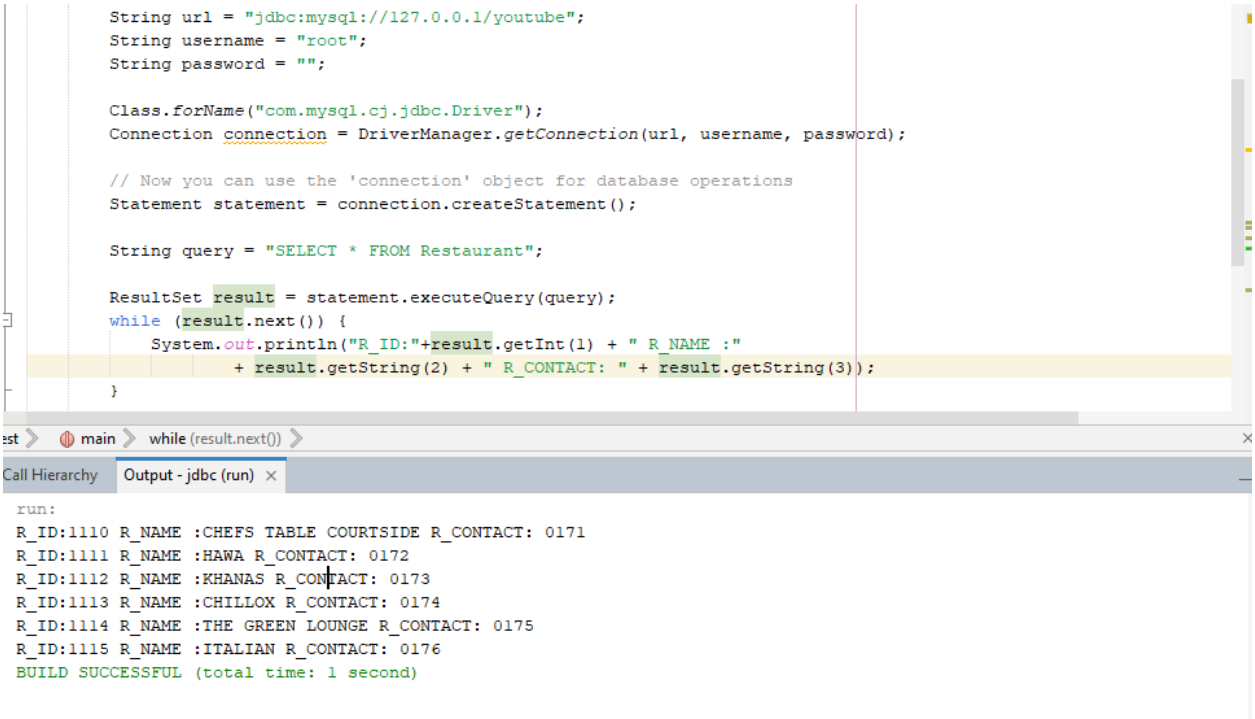


Fig5: NetBeans

Conclusion

In summary, the Restaurant Management System database project is a powerful tool that brings efficiency, accuracy, and enhanced customer satisfaction to restaurant operations. It contributes to the overall success and growth of the business by providing a well-organized and technology-driven approach to management.