

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

SL NO	TOPICS	PAGE NO
	Title Page	1
	Table of Content	2
1	Gantt Chart	3
2	Introduction	4
3	Case Study	4
4	E-R Diagram	5
5	Normalization	6-12
6	Finalization	13
7	Table Creation	13-18
8	Data Insertion	19-25
9	Query Test	26-38
10	Database Connection	39-42
10	Conclusion	42

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 a comprehensive 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 and staff 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 takes order. One waiter can 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 waiters.

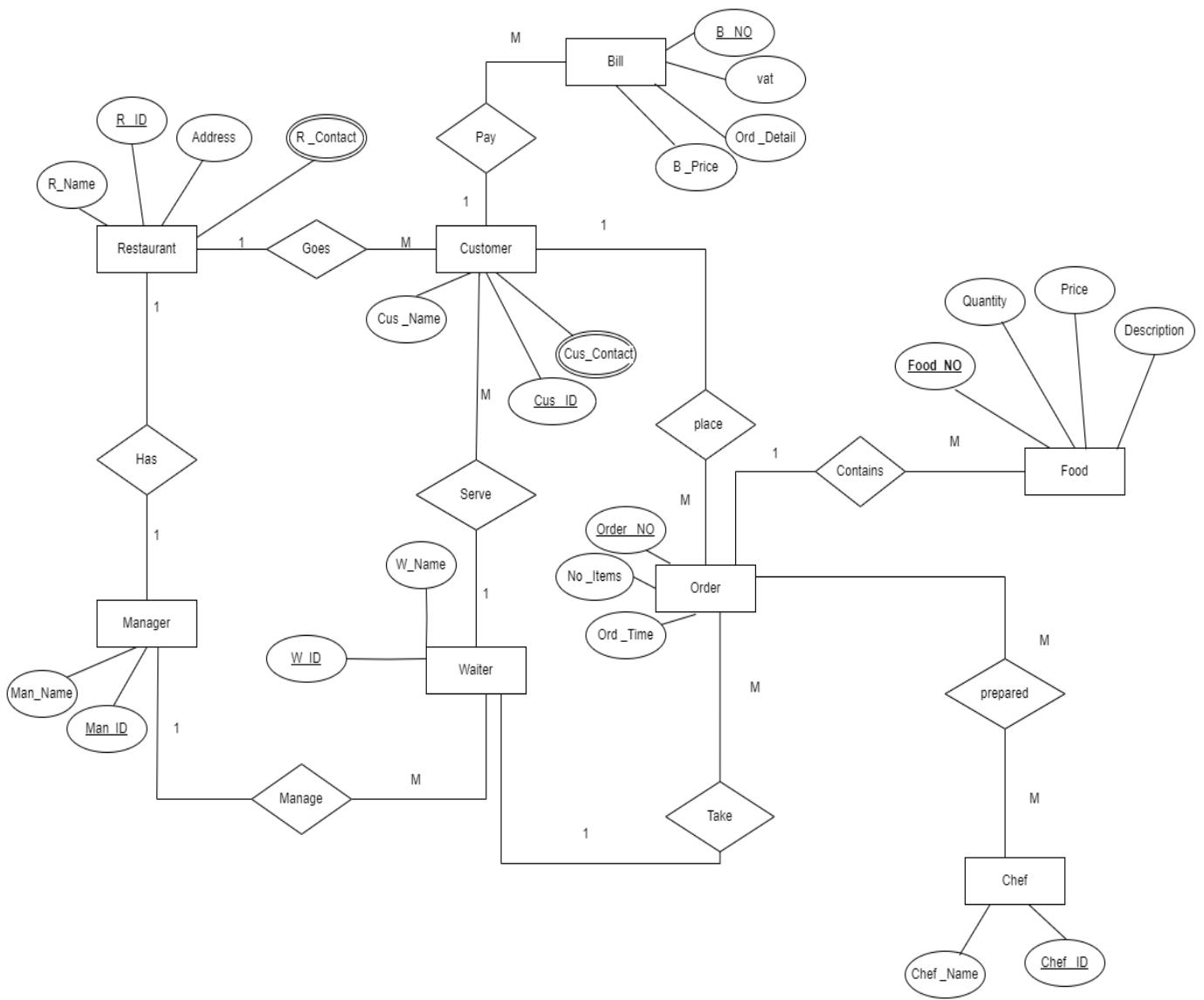


Fig 1: E-R Diagram for Restaurant Management System

Normalization:

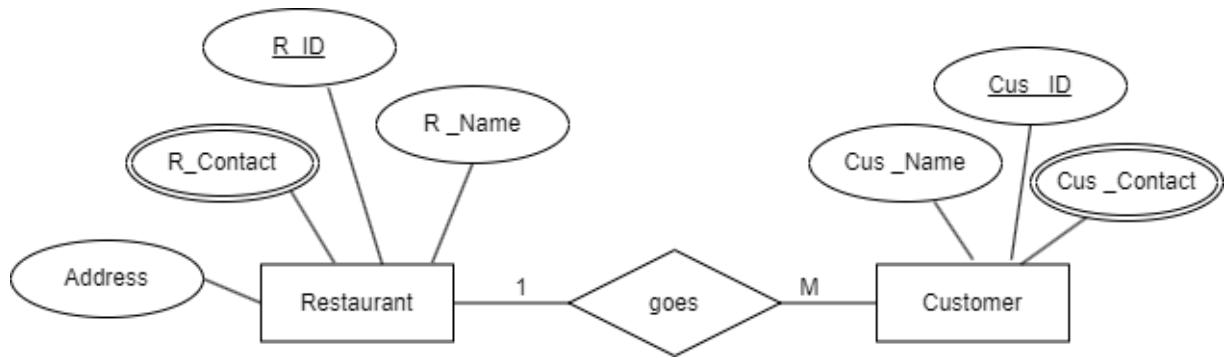


Fig 2: Relation between Restaurant and Customer

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

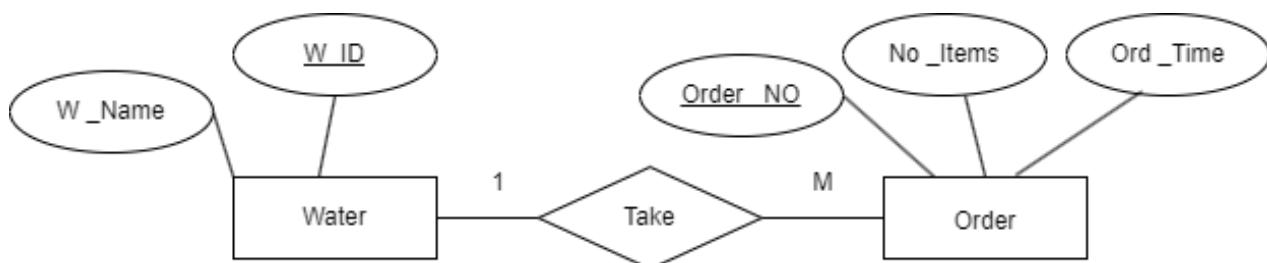


Fig 3: Relation between Waiter and Order

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

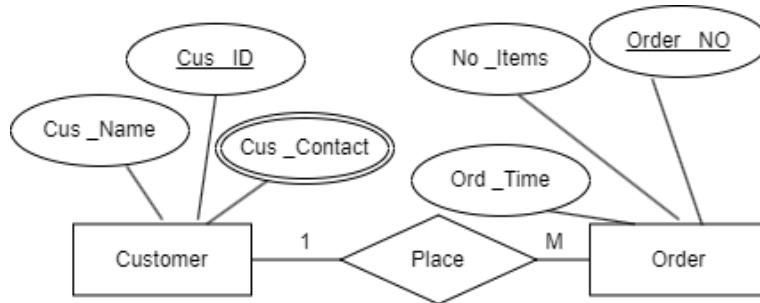


Fig 4: Relation between Customer and Order

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

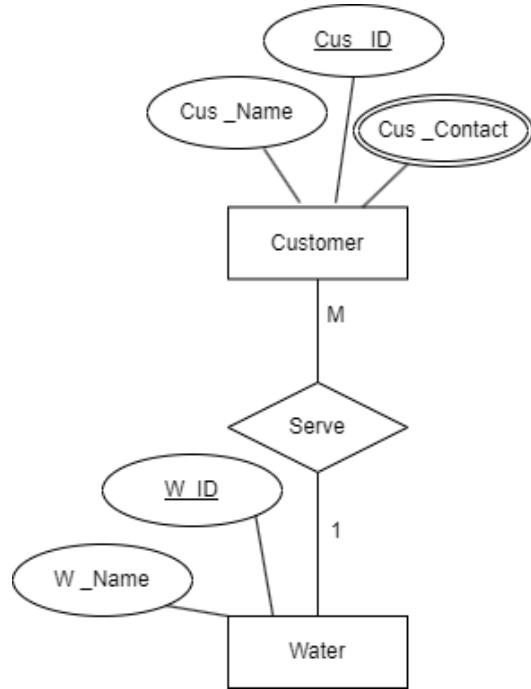


Fig 5: Relation between Customer and Waiter

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

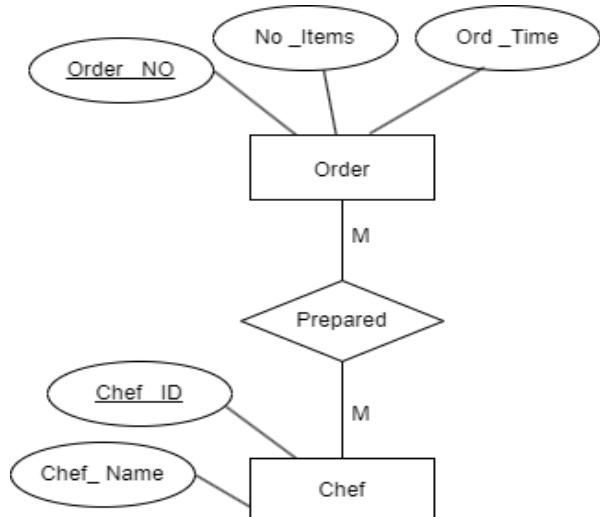


Fig 6: Relation between Order and Chef

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

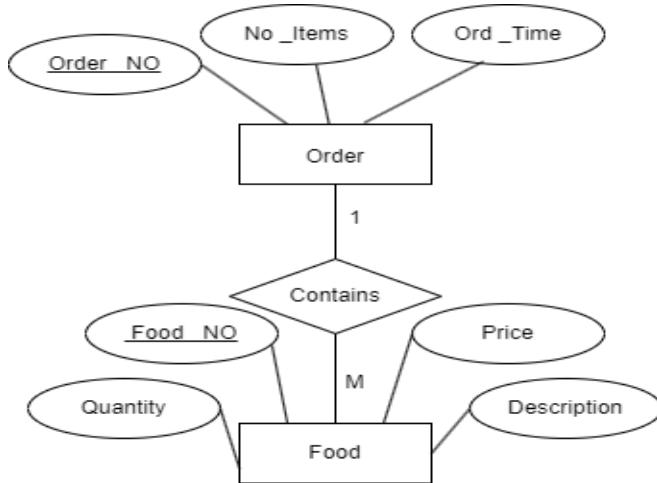


Fig 7: Relation between Order and Food

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)

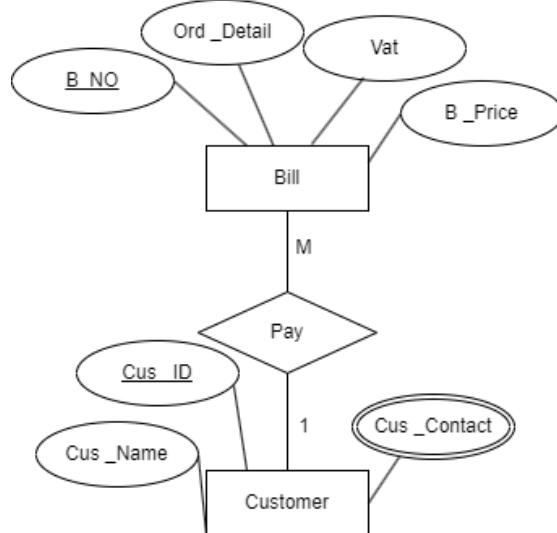


Fig 8: Relation between Customer and Bill

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_Price, Cus_ID(FK)

BillDescription: B_Price, Vat

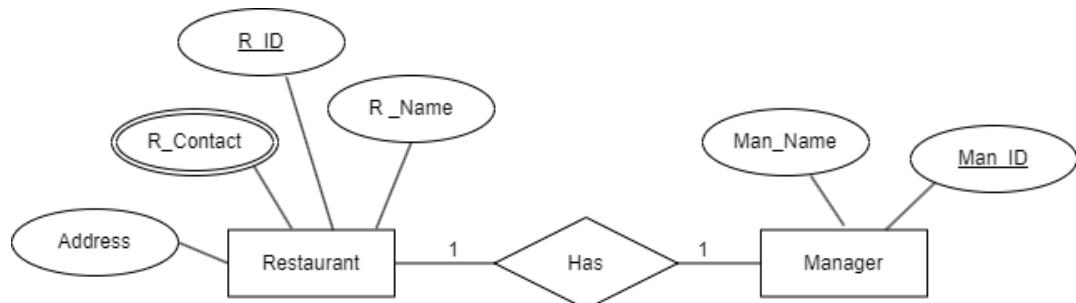


Fig 9: Relation between Restaurant and Manager

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

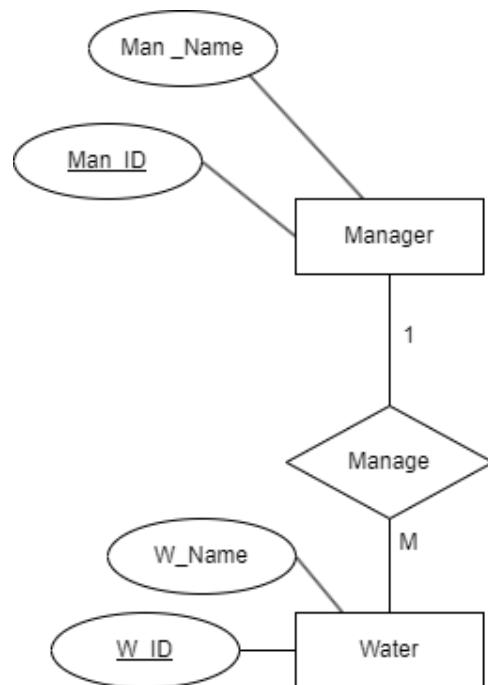


Fig 10: Relation between Manager and Waiter

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

Home > SQL > SQL Commands

Autocommit Display 10

```
CREATE TABLE Restaurant (
    R_ID NUMBER PRIMARY KEY,
    R_Name VARCHAR2(100),
    R_Contact VARCHAR2(20)
)
DESC Restaurant
```

Results Explain Describe Saved SQL History

Object Type TABLE Object RESTAURANT

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESTAURANT	R_ID	Number	-	-	-	1	-	-	-
	R_NAME	Varchar2	100	-	-	-	✓	-	-
	R_CONTACT	Varchar2	20	-	-	-	✓	-	-
									1 - 3

2. RestaurantINFO Table

Home > SQL > SQL Commands

Autocommit Display 10

```
CREATE TABLE RestaurantINFO (
    R_Name VARCHAR2(50) PRIMARY KEY,
    Address VARCHAR2(55)
)
DESC RestaurantINFO
```

Results Explain Describe Saved SQL History

Object Type TABLE Object RESTAURANTINFO

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESTAURANTINFO	R_NAME	Varchar2	50	-	-	1	-	-	-
	ADDRESS	Varchar2	55	-	-	-	✓	-	-
									1 - 2

3. Customer Table

Home > SQL > SQL Commands

Autocommit Display 10

```
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	Comment
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

```
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

```
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

```
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

```
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

```
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

```
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

```
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

```
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
BILDDESCRIPTION	PRICE	Number	-	-	-	1	-	-	
	VAT	Varchar2	15	-	-	-	✓	-	
									1 - 2

12. Manager Table

Home > SQL > SQL Commands

Autocommit Display 10

```
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	-	-	1 - 2
	MAN_NAME	Varchar2	50	-	-	-	✓	-	

13. ManagerDetails Table

Home > SQL > SQL Commands

Autocommit Display 10

```
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	-	-	1 - 2
	MAN_ID	Number	-	-	-	-	✓	-	

14. WaiterManage Table

Home > SQL > SQL Commands

Autocommit Display 10

```
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	-	-	1 - 3
	W_NAME	Varchar2	50	-	-	-	✓	-	
	MAN_ID	Number	-	-	-	-	✓	-	

Data Insertion

1. Value Insertion of table Restaurant.

Home > SQL > SQL Commands

Autocommit Display 10

```
INSERT INTO Restaurant (R_ID, R_Name, R_Contact) VALUES (1110, 'CHEFS TABLE COURTSIDE', '0171');
INSERT INTO Restaurant (R_ID, R_Name, R_Contact) VALUES (1111, 'HAWA', '0172');
INSERT INTO Restaurant (R_ID, R_Name, R_Contact) VALUES (1112, 'KHANAS', '0173');
INSERT INTO Restaurant (R_ID, R_Name, R_Contact) VALUES (1113, 'CHILLOX', '0174');
INSERT INTO Restaurant (R_ID, R_Name, R_Contact) VALUES (1114, 'THE GREEN LOUNGE', '0175');
INSERT INTO Restaurant (R_ID, R_Name, R_Contact) VALUES (1115, 'ITALIAN', '0176');
SELECT *FROM RESTAURANT
```

Results Explain Describe Saved SQL History

R_ID	R_NAME	R_CONTACT
1110	CHEFS TABLE COURTSIDE	0171
1111	HAWA	0172
1112	KHANAS	0173
1113	CHILLOX	0174
1114	THE GREEN LOUNGE	0175
1115	ITALIAN	0176

6 rows returned in 0.00 seconds [CSV Export](#)

2. Value Insertion of table Customer.

Home > SQL > SQL Commands

Autocommit Display 10

```
INSERT INTO Customer (Cus_ID, Cus_Name, Cus_Contact, R_ID) VALUES (301, 'DIBBO', '0161', 1110);
INSERT INTO Customer (Cus_ID, Cus_Name, Cus_Contact, R_ID) VALUES (302, 'MARUF', '0162', 1110);
INSERT INTO Customer (Cus_ID, Cus_Name, Cus_Contact, R_ID) VALUES (303, 'MIRAT', '0163', 1110);
INSERT INTO Customer (Cus_ID, Cus_Name, Cus_Contact, R_ID) VALUES (304, 'TONY', '0164', 1110);
INSERT INTO Customer (Cus_ID, Cus_Name, Cus_Contact, R_ID) VALUES (305, 'DIP', '0165', 1110);
INSERT INTO Customer (Cus_ID, Cus_Name, Cus_Contact, R_ID) VALUES (306, 'EMON', '0166', 1110);
SELECT *FROM CUSTOMER
```

Results Explain Describe Saved SQL History

CUS_ID	CUS_NAME	CUS_CONTACT	R_ID
302	MARUF	0162	1110
303	MIRAT	0163	1110
304	TONY	0164	1110
305	DIP	0165	1110
306	EMON	0166	1110
301	DIBBO	0161	1110

6 rows returned in 0.01 seconds [CSV Export](#)

3. Value Insertion of table Orderr.

Home > SQL > SQL Commands

Autocommit Display 10

```
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

```
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

```
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    
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    
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

```
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

```
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.

Autocommit Display 10 Save

```
SELECT PRICE, VAT FROM BillDescription WHERE PRICE BETWEEN 350 AND 800.
```

Results Explain Describe Saved SQL History

PRICE	VAT
350	10
600	20
800	20

3 rows returned in 0.00 seconds [CSV Export](#)

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

Autocommit Display 10 Save

```
SELECT R_ID, R_NAME FROM RESTAURANT WHERE R_CONTACT=0171
```

Results Explain Describe Saved SQL History

R_ID	R_NAME
1110	CHEFS TABLE COURTSIDE

1 rows returned in 0.00 seconds [CSV Export](#)

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]'.

Autocommit Display 10 Save

```
SELECT ' Order '||ORDER_NO || ' Has ' || DESCRIPTION AS Order_Description FROM FOOD
WHERE ORDER_NO IN (1, 2, 3);
```

Results Explain Describe Saved SQL History

R_ID	R_NAME
1110	CHEFS TABLE COURTSIDE

1 rows returned in 0.00 seconds [CSV Export](#)

- 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'.

Autocommit Display 10 Save Run

```
SELECT CUS_NAME, CUS_ID FROM CUSTOMER WHERE CUS_NAME LIKE '%D%O%' OR CUS_NAME LIKE '%M%'
```

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

CUS_NAME	CUS_ID
MARUF	302
MIRAT	303
EMON	306
DIBBO	301

4 rows returned in 0.01 seconds [CSV Export](#)

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

Autocommit Display 10 Save Run

```
SELECT FOOD_NO, QUANTITY, PRICE FROM FoodDetails WHERE FOOD_NO>=815 OR PRICE=350
```

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

FOOD_NO	QUANTITY	PRICE
800	3	350
815	5	600
816	6	800

3 rows returned in 0.02 seconds [CSV Export](#)

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

Autocommit Display 10 Save Run

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

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

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

8 rows returned in 0.00 seconds [CSV Export](#)

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

The screenshot shows a MySQL query interface with the following details:

- Autocommit:** Checked.
- Display:** Set to 10 rows.
- SQL Query:** `SELECT W_NAME, W_ID, ORDER_NO FROM Waiter ORDER BY W_ID ASC`
- Results:** A table with columns `W_NAME`, `W_ID`, and `ORDER_NO`. The data is:

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

- Timing:** 8 rows returned in 0.02 seconds.
- CSV Export:** Link available to download the results as CSV.

- 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?

The screenshot shows a MySQL query interface with the following details:

- Autocommit:** Checked.
- Display:** Set to 10 rows.
- SQL Query:** `SELECT FOOD_NO, QUANTITY, CONCAT (DESCRIPTION, ORDER_NO) AS ORDEREDFOODNAME FROM FOOD`
- Results:** A table with columns `FOOD_NO`, `QUANTITY`, and `ORDEREDFOODNAME`. The data is:

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

- Timing:** 6 rows returned in 0.02 seconds.
- CSV Export:** Link available to download the results as CSV.

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

The screenshot shows a MySQL query interface with the following details:

- Autocommit:** Checked.
- Display:** Set to 10 rows.
- SQL Query:** `SELECT R_NAME, R_ID FROM Restaurant WHERE SUBSTR(R_NAME,1,4) = 'HAWA'`
- Results:** A table with columns `R_NAME` and `R_ID`. The data is:

R_NAME	R_ID
HAWA	1111

- Timing:** 1 rows returned in 0.00 seconds.
- CSV Export:** Link available to download the results as CSV.

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

The screenshot shows a MySQL query editor interface. At the top, there are buttons for Autocommit (checked), Display (set to 10), Save, and Run. Below the buttons is a text input field containing the SQL query: `SELECT MAN_Name, LENGTH(MAN_NAME) AS LENGTH FROM MANAGER;`. Underneath the query is a results table with two columns: `MAN_NAME` and `LENGTH`. The data in the table is as follows:

MAN_NAME	LENGTH
RONY	4
JON	3
RAKIB	5
ESHAN	5
RONY	4
NIROB	5

Below the table, it says "6 rows returned in 0.01 seconds" and there is a link to "CSV Export".

- 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'.

The screenshot shows a MySQL query editor interface. At the top, there are buttons for Autocommit (checked), Display (set to 10), Save, and Run. Below the buttons is a text input field containing the SQL query: `SELECT MAN_ID, TRIM ('B'FROM'RAKIB') AS NEW_MANAGER_NAME FROM MANAGER WHERE MAN_NAME = 'RAKIB'`. Underneath the query is a results table with two columns: `MAN_ID` and `NEW_MANAGER_NAME`. The data in the table is as follows:

MAN_ID	NEW_MANAGER_NAME
2250	RAKI

Below the table, it says "1 rows returned in 0.00 seconds" and there is a link to "CSV Export".

- 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.

Home > SQL > SQL Commands

Autocommit Display 10 [Save](#) [Run](#)

```
SELECT B_NO, B_PRICE, ORD_DETAIL,
DECODE(ORD_DETAIL, 'TANDOORI_MUSHROOM', B_PRICE*1.31, 'PANEER_MASALA', B_PRICE*1.15)
AS REVERSED_FOODPRICE FROM BILL
```

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

B_NO	B_PRICE	ORD_DETAIL	REVERSED_FOODPRICE
10	600	FISH_CUTLET	-
12	350	PANEER_MASALA	402.5
14	250	TANDOORI_MUSHROOM	327.5
16	800	MALAI_KOFTA	-
18	800	MALAI_KOFTA	-
20	600	FISH_CUTLET	-

6 rows returned in 0.00 seconds [CSV Export](#)

- 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.'

Home > SQL > SQL Commands

Autocommit Display 10 [Save](#) [Run](#)

```
SELECT DESCRIPTION, REPLACE('FISH_CUTLET', 'CUTLET', 'FRI') AS MODIFY_FOOD_DESCRIPTION FROM FOOD WHERE DESCRIPTION='FISH_CUTLET'
```

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

DESCRIPTION	MODIFY_FOOD_DESCRIPTION
FISH_CUTLET	FISH_FRI

1 rows returned in 0.00 seconds [CSV Export](#)

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

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT W_ID, W_NAME, INSTR(W_NAME,'R') FROM WAITER;
```

Results Explain Describe Saved SQL History

W_ID	W_NAME	INSTR(W_NAME,'R')
1001	ROY	1
1002	RAJ	1
1003	NILOY	0
1004	KARIM	3
1005	SHANTO	0
1006	ROY	1
1007	JUHAB	0
1008	SAGAR	5

8 rows returned in 0.00 seconds [CSV Export](#)

Aggregate Query

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

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT MAX(B_PRICE) FROM BILL
```

Results Explain Describe Saved SQL History

MAX(B_PRICE)
800

1 rows returned in 0.00 seconds [CSV Export](#)

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

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT MIN(B_PRICE) FROM BILL
```

Results Explain Describe Saved SQL History

MIN(B_PRICE)
250

1 rows returned in 0.00 seconds [CSV Export](#)

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

Home > SQL > SQL Commands

Autocommit Display 10 [Save](#) [Run](#)

```
SELECT AVG(MAX(B_PRICE)) AS AVERAGE FROM BILL GROUP BY B_PRICE
```

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

AVERAGE
500

1 rows returned in 0.02 seconds [CSV Export](#)

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.

Home > SQL > SQL Commands

Autocommit Display 10 [Save](#) [Run](#)

```
SELECT AVG(PRICE), MAX(PRICE), MIN(PRICE), SUM(PRICE) FROM BILLDESCRIPTION WHERE VAT IN(5,20)
```

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

AVG(PRICE)	MAX(PRICE)	MIN(PRICE)	SUM(PRICE)
550	800	250	1650

1 rows returned in 0.00 seconds [CSV Export](#)

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.

The screenshot shows a MySQL query editor interface. The SQL command entered is:

```
SELECT W_NAME FROM Waiter WHERE ORDER_NO > (SELECT ORDER_NO FROM Waiter WHERE W_ID = 1004)
```

The results table displays the names of the waiters:

W_NAME
NILOY
ROY
SAGAR

Below the table, it says "3 rows returned in 0.00 seconds" and there is a "CSV Export" link.

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

The screenshot shows a MySQL query editor interface. The SQL command entered is:

```
SELECT Cus_ID FROM Customer WHERE R_ID IN (SELECT R_ID FROM Waiter WHERE W_Name = 'RAJ')
```

The results table displays the customer IDs:

CUS_ID
302
303
304
305
306
301

Below the table, it says "6 rows returned in 0.00 seconds" and there is a "CSV Export" link.

- 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.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
SELECT QUANTITY, MIN(PRICE) FROM FoodDetails GROUP BY QUANTITY HAVING MIN(PRICE)>(SELECT MIN(PRICE)
FROM FoodDetails WHERE FOOD_NO=800) ORDER BY MIN(PRICE)
```

Results Explain Describe Saved SQL History

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.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
SELECT FOOD_NO, DESCRIPTION FROM Food WHERE QUANTITY < ANY (SELECT QUANTITY FROM Food WHERE QUANTITY =
3) AND ORDER_NO <> 2|
```

Results Explain Describe Saved SQL History

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'.

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
SELECT W_NAME FROM Waiter WHERE Cus_ID = (SELECT Cus_ID FROM CUSTOMER WHERE CUS_NAME = 'DIBBO')|
```

Results Explain Describe Saved SQL History

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.

Home > SQL > SQL Commands

Autocommit

```
SELECT C.Cus_Name, C.Cus_Contact, R.R_Name FROM Customer C JOIN Restaurant R ON C.R_ID=R.R_ID
```

Results Explain Describe Saved SQL History

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.

Home > SQL > SQL Commands

Autocommit

```
SELECT W_ID, W_Name FROM Waiter W JOIN Food F ON W.ORDER_NO=F.ORDER_NO
```

Results Explain Describe Saved SQL History

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.

Home > SQL > SQL Commands

Autocommit

```
SELECT F.Food_NO, F.Quantity, F.Description, O.No_Items, O.Ord_Time FROM Food F LEFT OUTER JOIN OrderINFO O ON F.Quantity = O.No_Items;
```

Results Explain Describe Saved SQL History

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

```
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.

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

- Autocommit:** Checked.
- Display:** Set to 100000.
- Buttons:** Save and Run.
- SQL Query:**

```
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 Tab:** Active tab.
- Table Data:** A grid showing the results of the query. The columns are WAITER_ID, WAITER_NAME, CUSTOMER_ID, and ORDER_NO. The data is as follows:

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

- Timing:** 8 rows returned in 0.00 seconds.
- Export:** CSV Export link.

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.

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

Autocommit: On
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

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:

The screenshot shows the NetBeans IDE interface. In the top editor window, a Java file named `test2Table` contains the following code:

```
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 create a table named 'Restaurant'
17        String createTableQuery = "CREATE TABLE Restaurant (" +
18            + "R_ID INT PRIMARY KEY," +
19            + "R_Name VARCHAR(100)," +
20            + "R_Contact VARCHAR(20))";
21
22        statement.executeUpdate(createTableQuery);
23        System.out.println("Table 'Restaurant' created successfully!");
24    } catch (SQLException ex) {
25        ex.printStackTrace();
26    }
}
```

In the bottom output window, the results of the run command are shown:

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

Fig1: NetBeans

The screenshot shows the phpMyAdmin interface. At the top, a message says "Your SQL query has been executed successfully." Below it, the SQL query is displayed:

```
desc Restaurant;
```

Below the query, the table structure is shown in a table:

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	

At the bottom, there is a "Query results operations" section.

Fig2: phpMyAdmin

Data Insertion:

The screenshot shows the NetBeans IDE interface. In the top editor pane, a Java class named `test2table` is displayed. The `main` method contains code to connect to a MySQL database and execute an `INSERT` query. The `insertTableQuery` string is highlighted in yellow. In the bottom output pane, the results of the run are shown, indicating that values were inserted successfully into the `'Restaurant'` table.

```
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
17            String insertTableQuery = "INSERT INTO Restaurant (R_ID, R_Name, R_Contact) "
18                + "VALUES (1110, 'CHEFS TABLE COURTSIDE', '0171')";
19
20            // Execute the query to insert values
21        }
22    }
23 }
```

Output - jdbc (run)

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

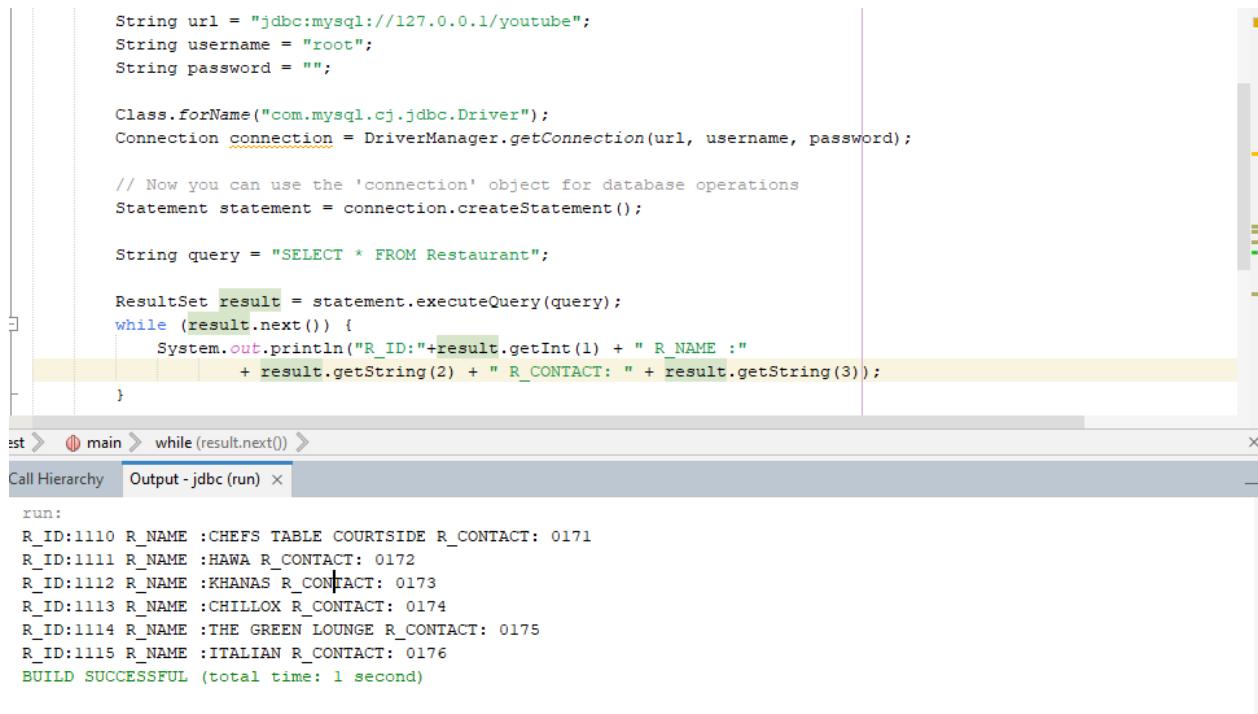
Fig3: NetBeans

The screenshot shows the phpMyAdmin interface with the `Restaurant` table selected. The table has four columns: `R_ID`, `R_Name`, and `R_Contact`. The data consists of six rows, each with a checkbox for selection and edit/delete buttons. The rows are:

	R_ID	R_Name	R_Contact
<input type="checkbox"/>	1110	CHEFS TABLE COURTSIDE	0171
<input type="checkbox"/>	1111	HAWA	0172
<input type="checkbox"/>	1112	KHANAS	0173
<input type="checkbox"/>	1113	CHILLOX	0174
<input type="checkbox"/>	1114	THE GREEN LOUNGE	0175
<input type="checkbox"/>	1115	ITALIAN	0176

Fig4: phpMyAdmin

Show full table:



The screenshot shows the NetBeans IDE interface. The code editor displays Java code for connecting to a MySQL database and printing the results of a SELECT query. The output window shows the printed data.

```
String url = "jdbc:mysql://127.0.0.1/youtube";
String username = "root";
String password = "";

Class.forName("com.mysql.cj.jdbc.Driver");
Connection connection = DriverManager.getConnection(url, username, password);

// Now you can use the 'connection' object for database operations
Statement statement = connection.createStatement();

String query = "SELECT * FROM Restaurant";

ResultSet result = statement.executeQuery(query);
while (result.next()) {
    System.out.println("R_ID:" + result.getInt(1) + " R_NAME :"
        + result.getString(2) + " R_CONTACT: " + result.getString(3));
}
```

Output - jdbc (run)

```
run:
R_ID:1110 R_NAME :CHEFS TABLE COURTSIDE R_CONTACT: 0171
R_ID:1111 R_NAME :HAWA R_CONTACT: 0172
R_ID:1112 R_NAME :KHANAS R_CONTACT: 0173
R_ID:1113 R_NAME :CHILLOX R_CONTACT: 0174
R_ID:1114 R_NAME :THE GREEN LOUNGE R_CONTACT: 0175
R_ID:1115 R_NAME :ITALIAN R_CONTACT: 0176
BUILD SUCCESSFUL (total time: 1 second)
```

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.