GARMENTS MANAGEMENT SYSTEM

Group Members:

Name	ID	Contribution
Mahmud Hasan	20-42202-1	User interface, Table creation, ER Diagram,
Rabbi Hasan Himel	20-42204-1	Use case, Activity, Relational Algebra, Schema Diagram, Table creation, Query writing
Mohi Uddin Anando	20-43585-1	Query writing, Class diagram, Normalization, Data insertion, Schema diagram
MONAYEM HASAN	20-42222-1	Introduction, ER Diagram, Data insertion
SALMAN	20-42395-1	Project Proposal, Scenario Description, Conclusion

Advance Database Management System (Section-E)

Contents

1. Introduction

The Garment Management System is a software application that automates the garment manufacturing and management process. It integrates with Oracle 10g database and uses a PHP interface to offer an end-to-end solution for garment manufacturing, inventory management, production tracking, employee roles and salary grades. This system simplifies the operations for garment manufacturers and managers by increasing productivity, reducing lead times and improving quality, all while managing costs. The integration with Oracle 10g database ensures secure data management, while the PHP interface provides an easy-to-use and flexible user interface. Ultimately, Garment Management System on Oracle 10g with PHP interface is a comprehensive solution for garment manufacturers and managers to streamline their operations and manage their business effectively.

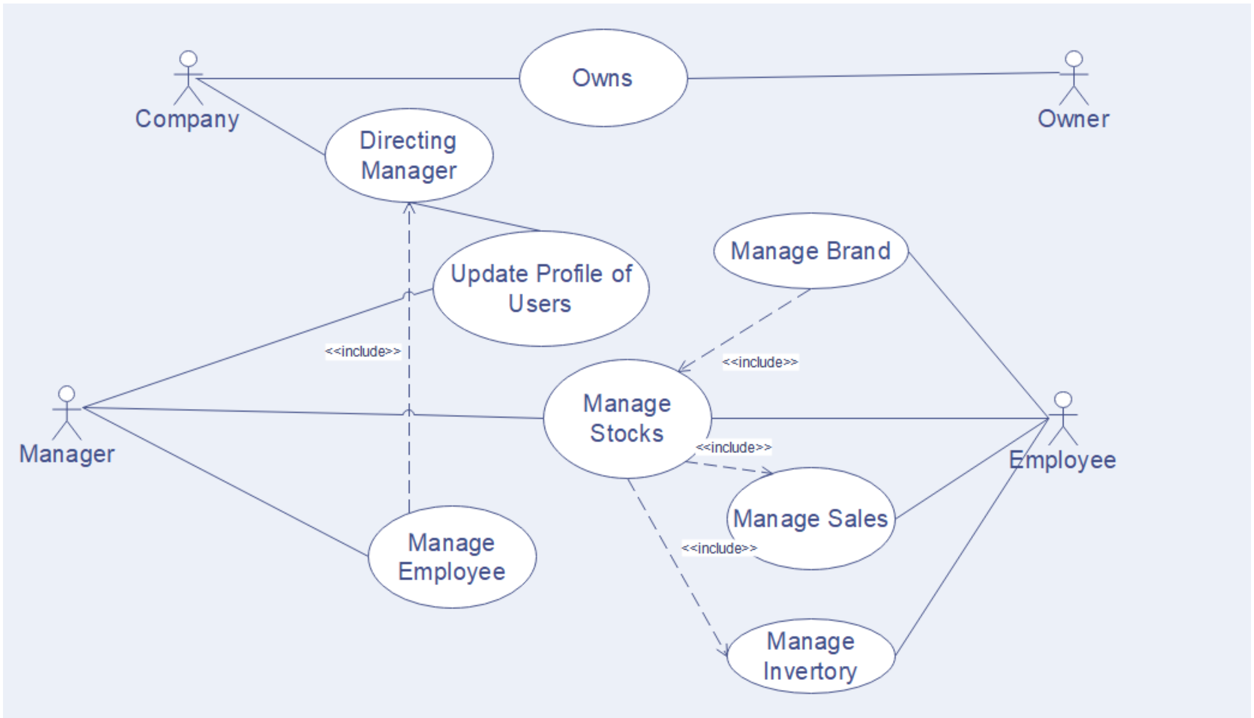
2. Project Proposal of Garment Management System

The Garment Management System is a software application designed to automate and streamline the entire garment manufacturing process, from order placement to delivery management. The system aims to simplify and enhance the daily operations of garment manufacturers and managers, ultimately resulting in increased productivity and reduced costs. The proposed system will be built using modern technologies such as Oracle 10g database and PHP interface, providing a robust and secure platform for managing data. The application will have an easy-to-use interface that makes it simple for users to access and manage data. The system will offer various features, including manager details, inventory management, stocks tracking, quality control, brands details, and delivery management. Users can place orders and track their status in real-time. The system will also provide real-time reporting capabilities that enable users to generate reports on inventory, stocks, and quality control. The Garment Management System is expected to increase productivity and reduce lead times by automating the entire process, from order placement to delivery management. By centralizing data, the system will improve communication and collaboration among team members, resulting in improved quality and reduced costs. In summary, the proposed Garment Management System is a comprehensive solution that simplifies and streamlines the garment manufacturing process.

The system is expected to benefit garment manufacturers and managers by increasing productivity, reducing lead times, improving quality, and lowering costs.

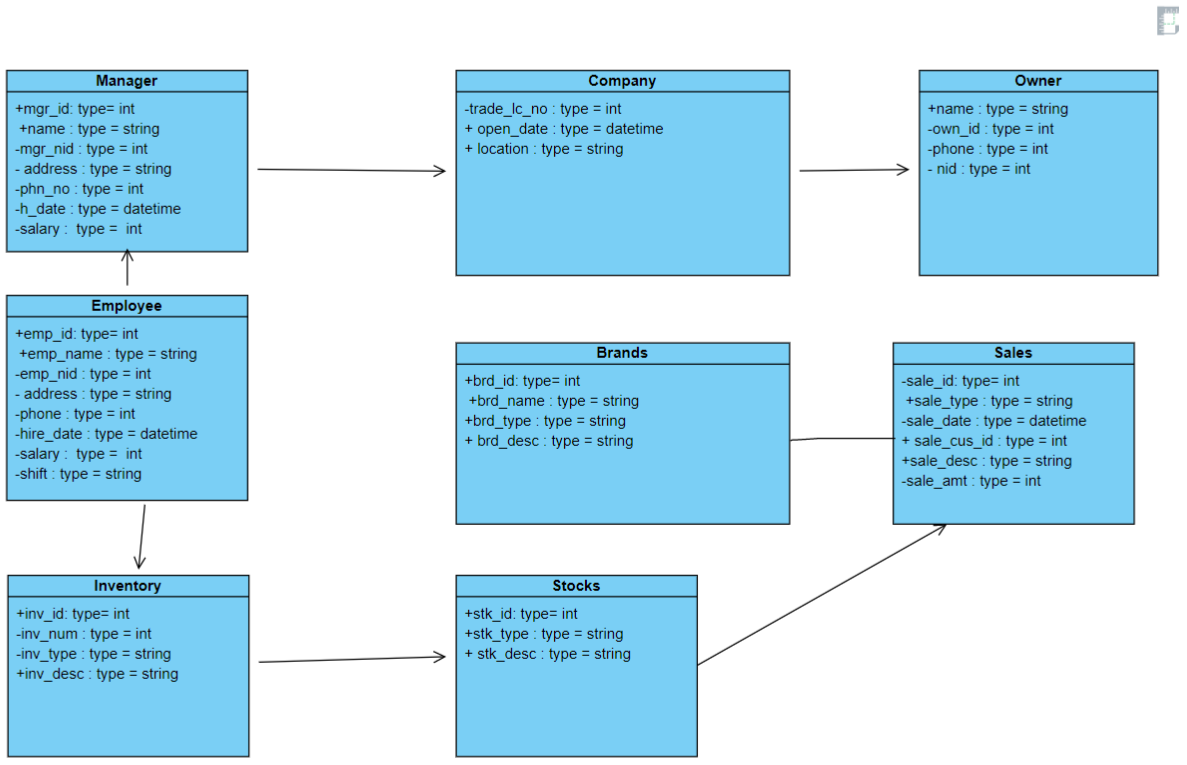
3. USE CASE, CLASS, ACTIVITY DIAGRAM

<u>USE CASE:</u>



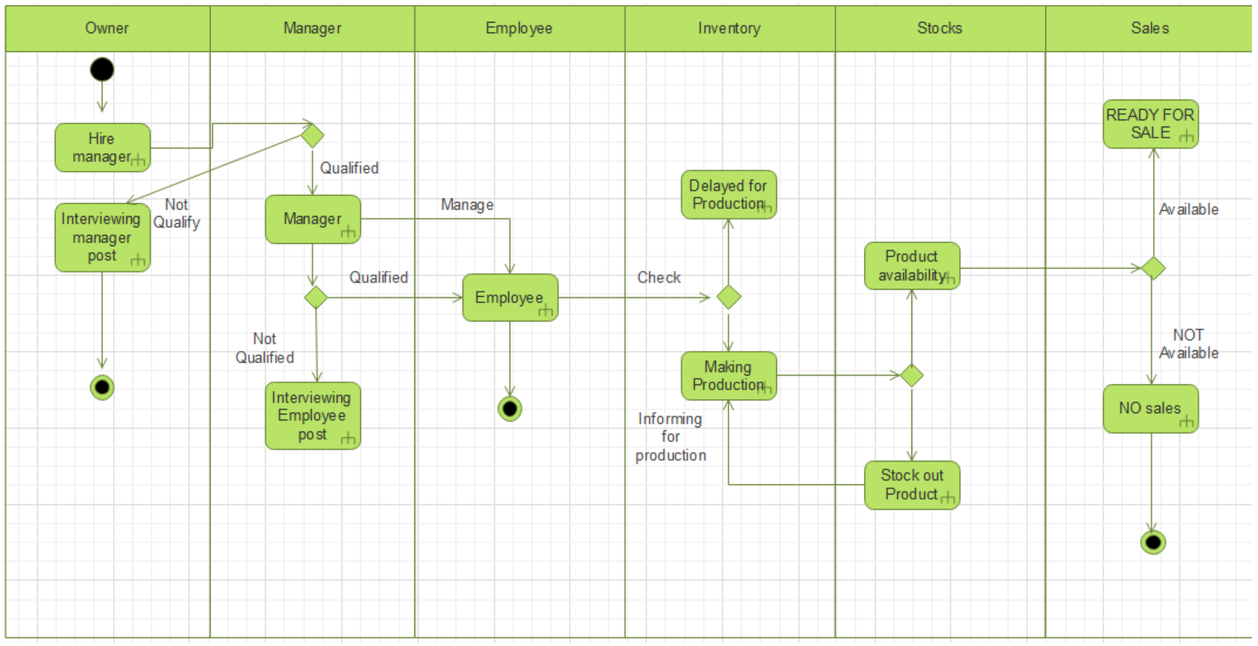
Use Case of Garments Management System

CLASS DIAGRAM:

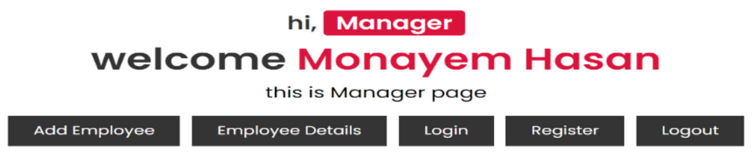


Manager
+mgr_id: type= int
+name : type = string
-mgr_nid : type = int
- address : type = string
-phn_no : type = int
-h_date : type = datetime
-salary : type = int
Company
-trade_lc_no : type = int
+ open_date : type = datetime
+ location : type = string
Owner
+name : type = string
-own_id : type = int
-phone : type = int
- nid : type = int
Employee
+emp_id: type= int
+emp_name : type = string
-emp_nid : type = int
- address : type = string
-phone : type = int
-hire_date : type = datetime
-salary : type = int
-shift : type = string
Brands
+brd_id: type= int
+brd_name : type = string
+brd_type : type = string
+ brd_desc : type = string
Sales
-sale_id: type= int
+sale_type : type = string
-sale_date : type = datetime
+ sale_cus_id : type = int
+sale_desc : type = string
-sale_amt : type = int
Inventory
+inv_id: type= int
-inv_num : type = int
-inv_type : type = string
+inv_desc : type = string
Stocks
+stk_id: type= int
+stk_type : type = string
+ stk_desc : type = string

Activity Diagram

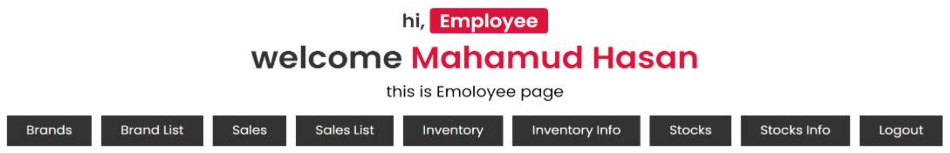


4. USER INTERFACE of Garment Management System


hi, Manager

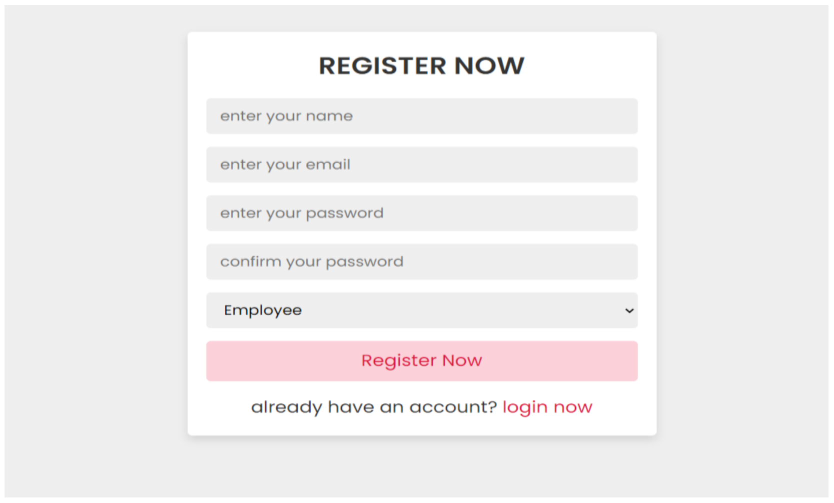
welcome Monayem Hasan

this is Manager page

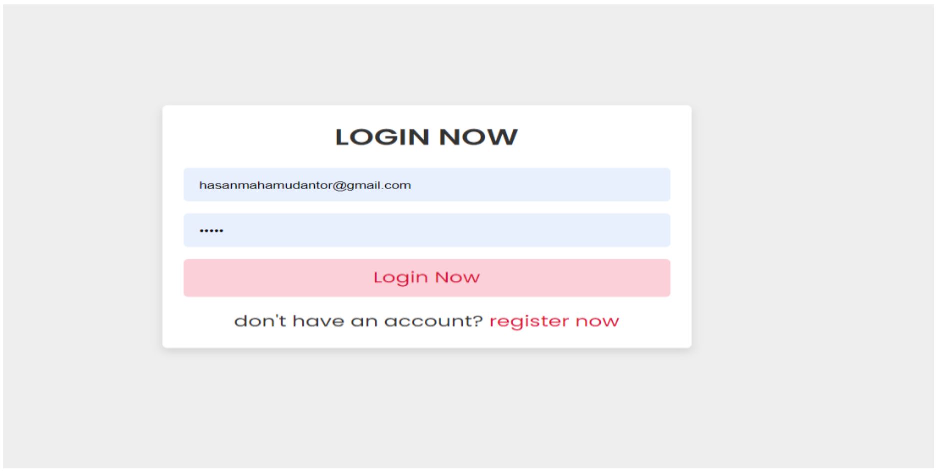

hi, Employee

welcome Mahamud Hasan

this is Emoloyee page


REGISTER NOW

already have an account? login now



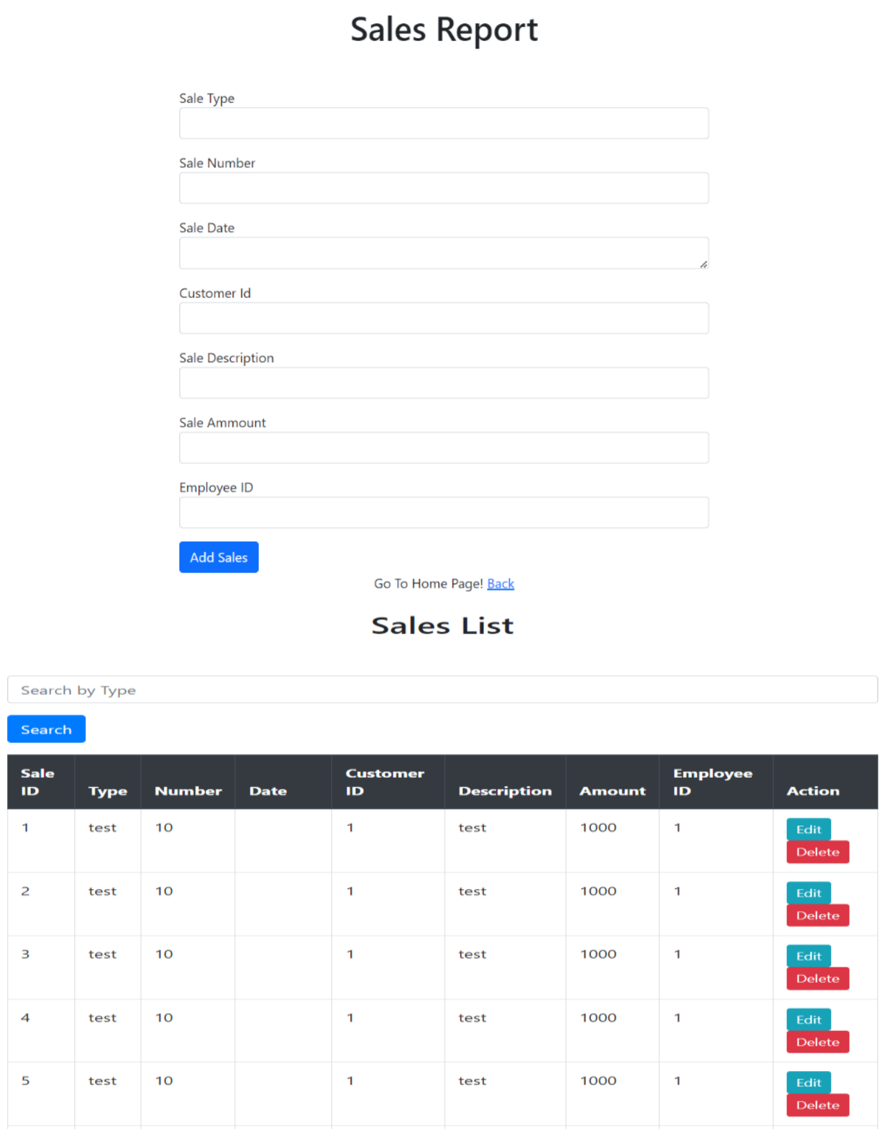
LOGIN NOW

hasanmahamudantor@gmail.com

•••••

Login Now

don't have an account? register now



Sales Report

Sale Type

Sale Number

Sale Date

Customer Id

Sale Description

Sale Ammount

Employee ID

Add Sales

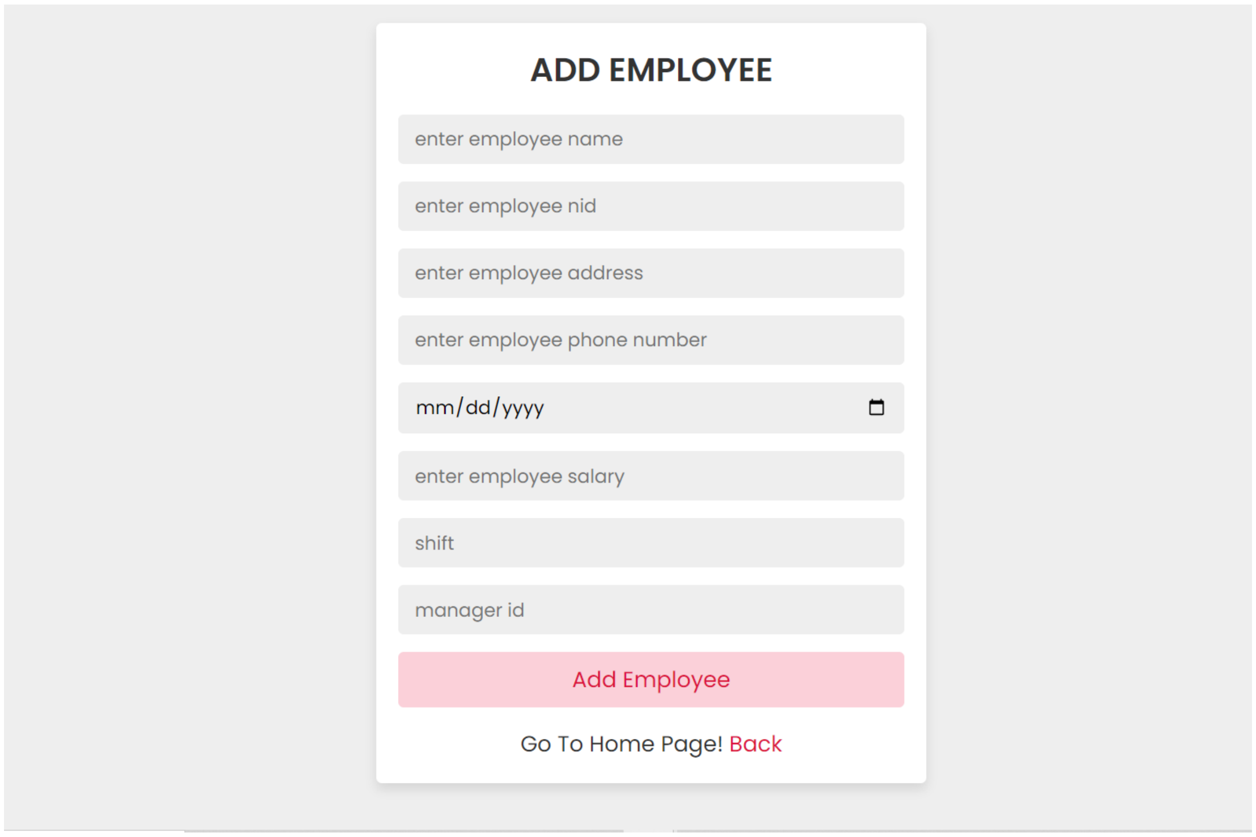
Go To Home Page! Back

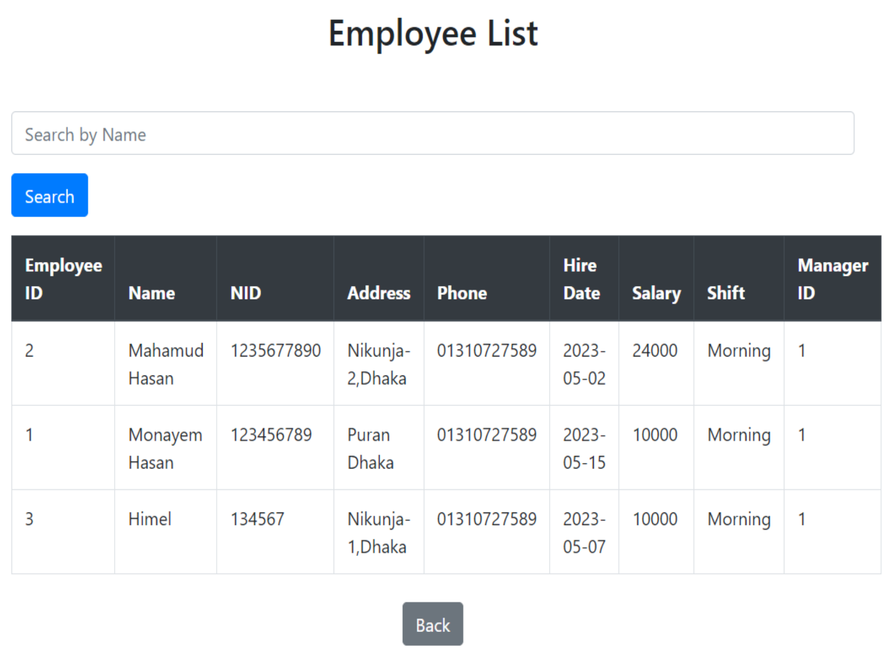
Sales List

Search by Type

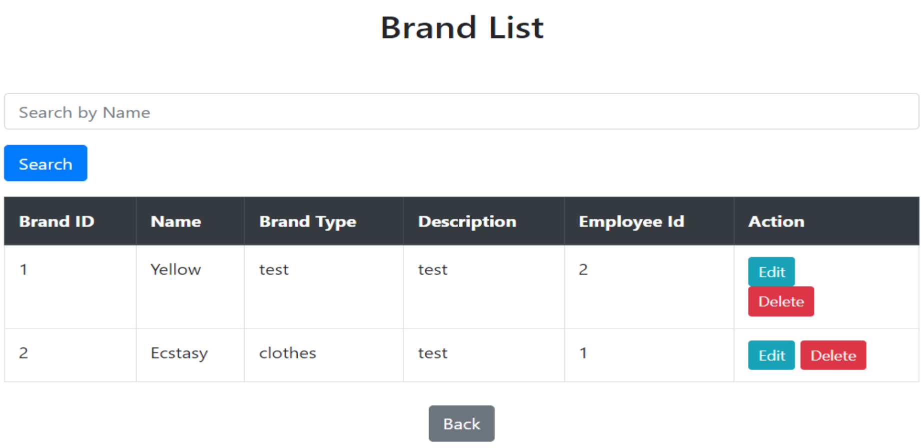
Search

Sale ID	Type	Number	Date	Customer ID	Description	Amount	Employee ID	Action
1	test	10		1	test	1000	1	Edit Delete
2	test	10		1	test	1000	1	Edit Delete
3	test	10		1	test	1000	1	Edit Delete
4	test	10		1	test	1000	1	Edit Delete
5	test	10		1	test	1000	1	Edit Delete




Employee List

Employee ID	Name	NID	Address	Phone	Hire Date	Salary	Shift	Manager ID
2	Mahamud Hasan	1235677890	Nikunja-2,Dhaka	01310727589	2023-05-02	24000	Morning	1
1	Monayem Hasan	123456789	Puran Dhaka	01310727589	2023-05-15	10000	Morning	1
3	Himel	134567	Nikunja-1,Dhaka	01310727589	2023-05-07	10000	Morning	1



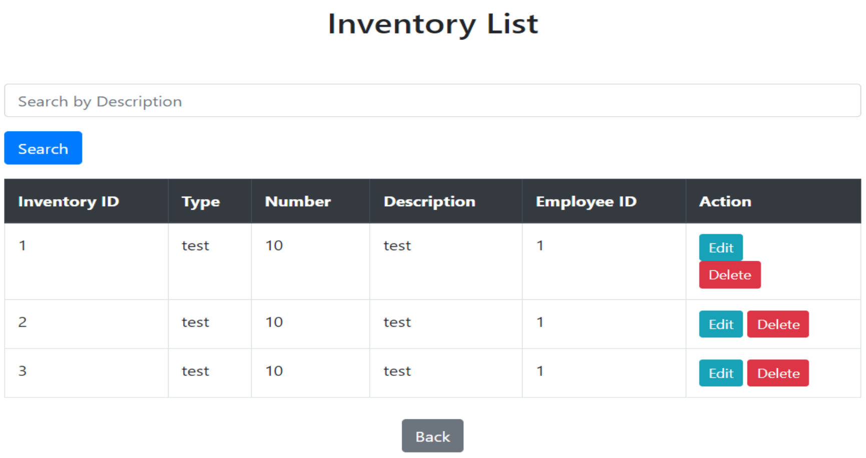
Brand List

Search by Name

Search

Brand ID	Name	Brand Type	Description	Employee Id	Action
1	Yellow	test	test	2	Edit Delete
2	Ecstasy	clothes	test	1	Edit Delete

Back



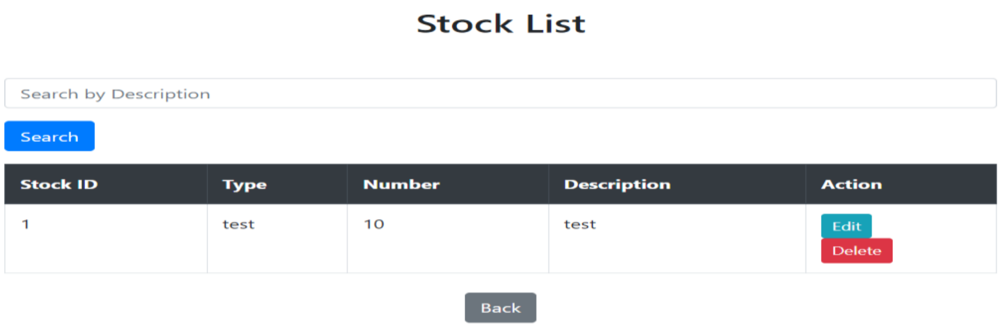
Inventory List

Search by Description

Search

Inventory ID	Type	Number	Description	Employee ID	Action
1	test	10	test	1	Edit Delete
2	test	10	test	1	Edit Delete
3	test	10	test	1	Edit Delete

Back



Stock List

Search by Description

Search

Stock ID	Type	Number	Description	Action
1	test	10	test	Edit Delete

Back

5. Scenario Description of Garment Management System

The garments company is set to open its doors soon, with all the necessary paperwork in order. It has obtained a trade license with a unique license number to operate legally. The company's location serves as its physical base, where customers can visit and browse the products. Interestingly, the company is owned by multiple individuals who each have their own owner ID, name, phone number, and national identification number (NID).

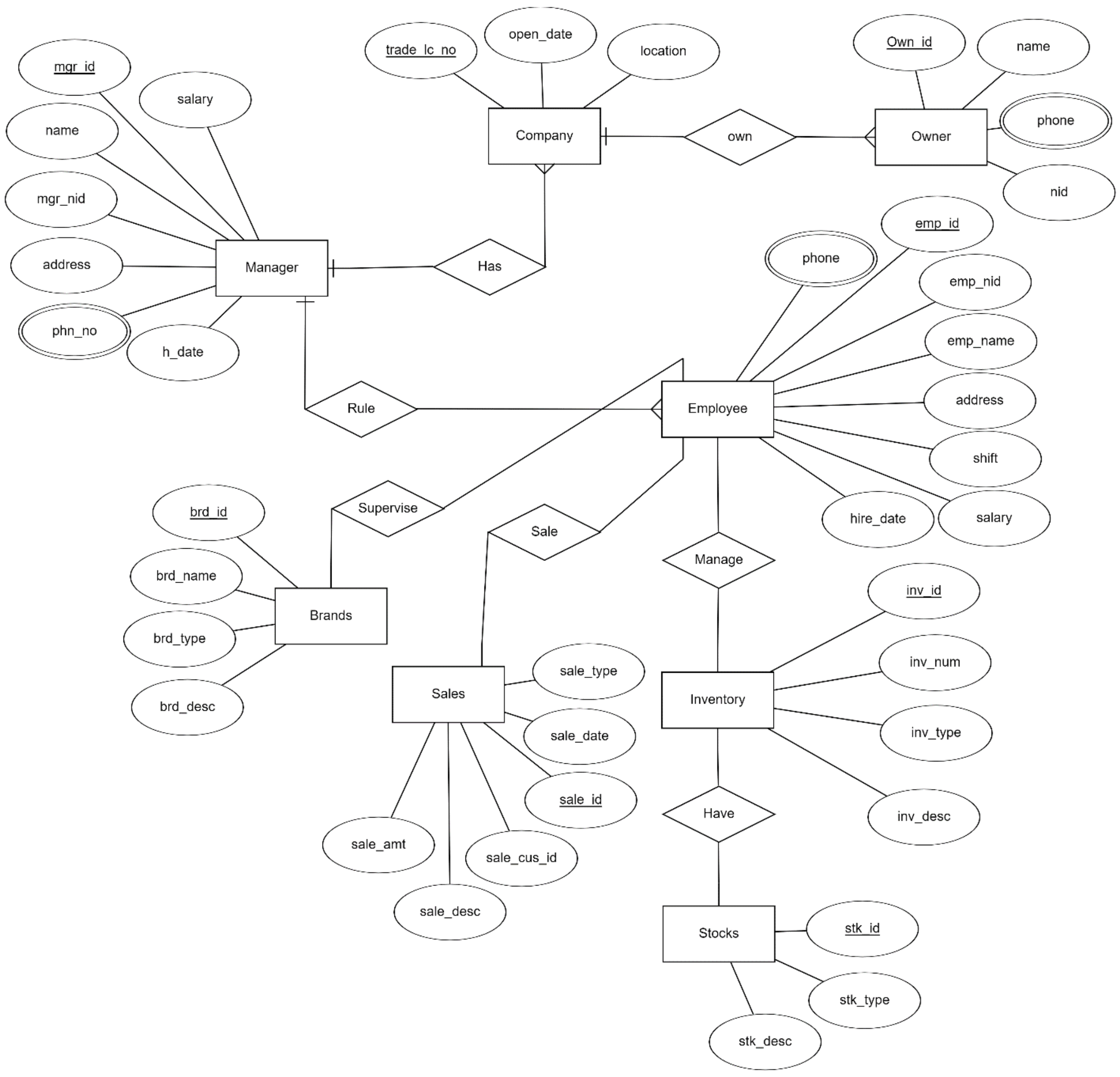
To oversee the operations of the company, a capable manager has been appointed. The manager takes care of multiple companies simultaneously, demonstrating excellent management skills. The manager possesses essential details such as a manager ID, name, salary, hire date, address, and manager NID.

The manager is responsible for a team of dedicated employees who work under their supervision. These employees play various roles within the company, and each employee has an employee ID, national identification number (NID), phone number, name, address, shift timing, salary, and hire date.

Among their responsibilities, the employees manage the inventory, which includes inventory numbers, types, descriptions, and IDs. Additionally, the inventory consists of different stocks, each identified by a stock ID, type, and description.

Furthermore, the employees also serve as sellers, engaging in sales activities. They handle various sale types, sale descriptions, sale IDs, sale amounts, and sale customer descriptions. Additionally, the employees oversee different brands, which are categorized by brand IDs, types, descriptions, and names.

6. ER Diagram of Garment Management System



7. Normalization

Has:

UNF: mgr_id, name, nid, address, phn_no, h_date, salary, trade_lc_no, open_date, location

1NF:

1. There is a multi-valued attribute which is phn_no.
2. mgr_id, name, nid, address, h_date, salary, trade_lc_no, open_date, location

2NF:

1. mgr_id, name, nid, address, h_date, salary
2. trade_lc_no, open_date, location
3. phn_no, mgr_id

3NF:

1. mgr_id, name, nid, address, h_date, salary
2. trade_lc_no, open_date, location
3. phn_no, mgr_id

Table Creation:

1. phn_no, mgr_id
2. mgr_id, name, nid, address, h_date, salary
3. **trade_lc_no,** open_date, location

Own:

UNF: trade_lc_no, open_date, location, name, phone, nid, own_id

1NF: There is a multi-valued attribute which is Phone.

1. trade_lc_no, open_date, location, name, nid, own_id

2NF:

1. own_id, name, nid
2. trade_lc_no, open_date, location
3. own_id, phone

3NF:

1. **own_id**, name, nid
2. trade_li_no, open_date, location
3. own_id, phone

Table Creation:

1. **own_id,** phone
2. own_id, name, nid
3. trade_li_no, open_date, location

Rule:

UNF: mgr_id, name, nid, address, phn_no, h_date, salary,emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date, phone

1NF:

1. mgr_id, name, nid, address, h_date, salary, phn_no, emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date

2NF:

1. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
2. mgr_id, name, nid, address, h_date, salary
3. emp_id, Phone
4. mgr_id, phn_no

3NF:

1. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
2. mgr_id, name, nid, address, h_date, salary
3. emp_id, Phone
4. mgr_id, phn_no

Table Creation:

1. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
2. mgr_id, name, nid, address, h_date, salary
3. emp_id, Phone
4. **mgr_id**, phn_no

Supervise:

UNF: brd_id, brd_name, brd_type, brd_desc, emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date, phone

1NF:

1. brd_id, brd_name, brd_type, brd_desc, emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date

2NF:

1. brd_id, brd_name, brd_type, brd_desc
2. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
3. emp_id, phone

3NF:

1. brd_id, brd_name, brd_type, brd_desc
2. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
3. emp_id, phone

Table Creation:

1. brd_id, brd_name, brd_type, brd_desc
2. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
3. **emp_id**, phone

Sale:

UNF: sale_id , sale_date, sale_type, sale_cus_id, sale_desc, sale_amt, emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date, phone

1NF:

1. sale_id, sale_date, sale_type, sale_cus_id, sale_desc, sale_amt, emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date

2NF:

1. sale_id, sale_date, sale_type, sale_cus_id, sale_desc, sale_amt
2. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
3. emp_id, phone

3NF:

1. sale_id, sale_date, sale_type, sale_cus_id, sale_desc, sale_amt
2. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
3. emp_id, phone

Table Creation:

1. **sale_id**, sale_date, sale_type, sale_cus_id, sale_desc, sale_amt
2. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
3. emp_id, phone

Have:
UNF: stk_id, stk_type, stk_desc, inv_id, inv_num. inv_type, inv_desc
1NF:
1. stk_id, stk_type, stk_desc, inv_id, inv_num. inv_type, inv_desc

2NF:
1. stk_id, stk_type, stk_desc
2. inv_id, inv_num. inv_type, inv_desc

3NF:
1. stk_id, stk_type, stk_desc
2. inv_id, inv_num. inv_type, inv_desc

Table Creation:
1. **stk_id**, stk_type, stk_desc
2. inv_id, inv_num. inv_type, inv_desc

Inventory:
UNF: inv_id, inv_num. inv_type, inv_desc, emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date, phone
1NF:
1. inv_id, inv_num. inv_type, inv_desc, emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date

2NF:
1. inv_id, inv_num. inv_type, inv_desc
2. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
3. emp_id, phone

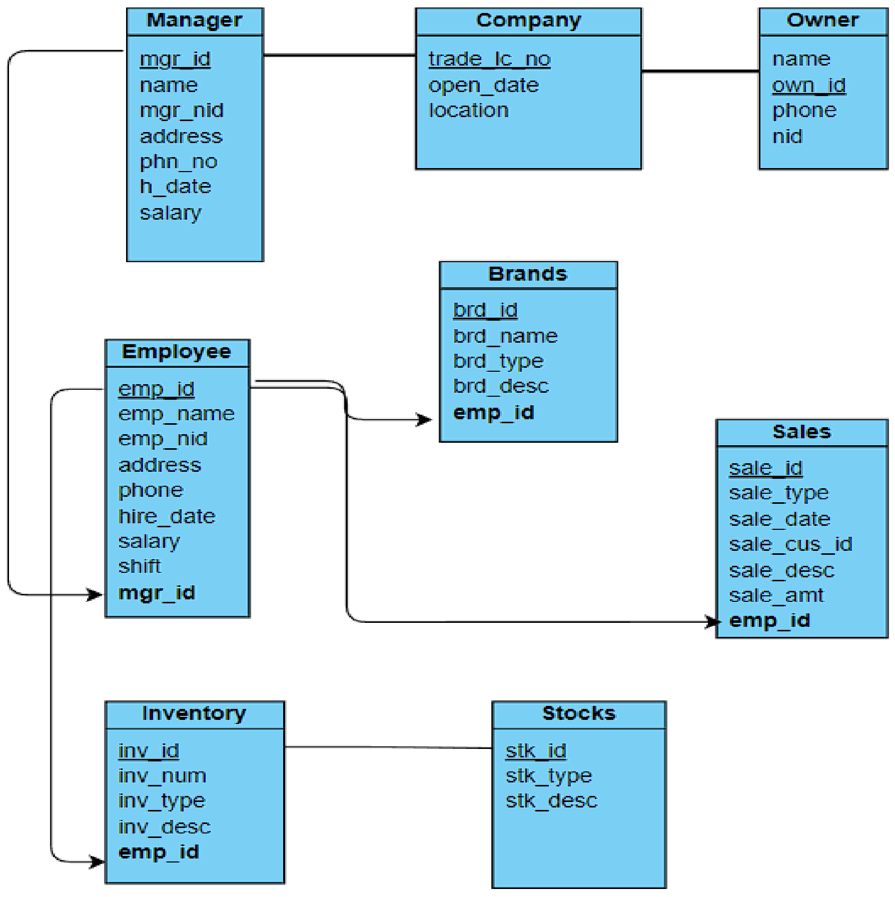
3NF:
1. inv_id, inv_num. inv_type, inv_desc
2. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
3. emp_id, phone

Table Creation:
1. inv_id, inv_num. inv_type, inv_desc
2. emp_id, emp_nid, emp_name, address, shift, emp_salary, hire_date
3. **emp_id**, phone

Final Tables:

1. **<u>trade_lc_no</u>**, open_date, location
2. **<u>own_id</u>**, name, phone, nid
3. **<u>mgr_id</u>**, name, mgr_nid, address, phone, h_date, salary
4. emp_id, emp_name, emp_nid, address, phone, h_date, salary, shift, **mgr_id**
5. **<u>inv_id</u>**, inv_type, inv_num, inv_desc, **emp_id**
6. **<u>stk_id</u>,** stk_type, stk_num, stk_desc
7. <u>brd_id</u>, brd_name, brd_type, brd_desc, **emp_id**
8. **<u>sale_id</u>**, sale_type, sale_num, sale_date, sale_cus_id, sale_desc, sale_amt, **emp_id**

8. Schema Diagram



9. Table Creation

CREATE TABLE:

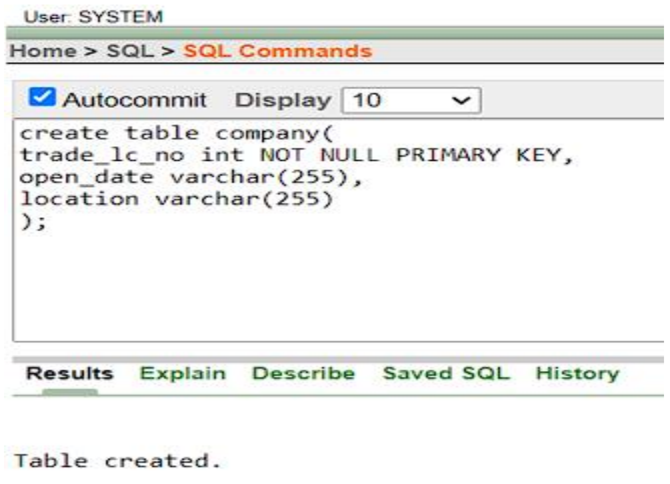
Create Table:

create table company(

trade_lc_no int NOT NULL PRIMARY KEY,

open_date varchar(255),

location varchar(255))



User: SYSTEM
Home > SQL > SQL Commands
Autocommit Display 10
Results Explain Describe Saved SQL History

```
create table company(
trade_lc_no int NOT NULL PRIMARY KEY,
open_date varchar(255),
location varchar(255)
);
```

```
Table created.
```

create table owner(

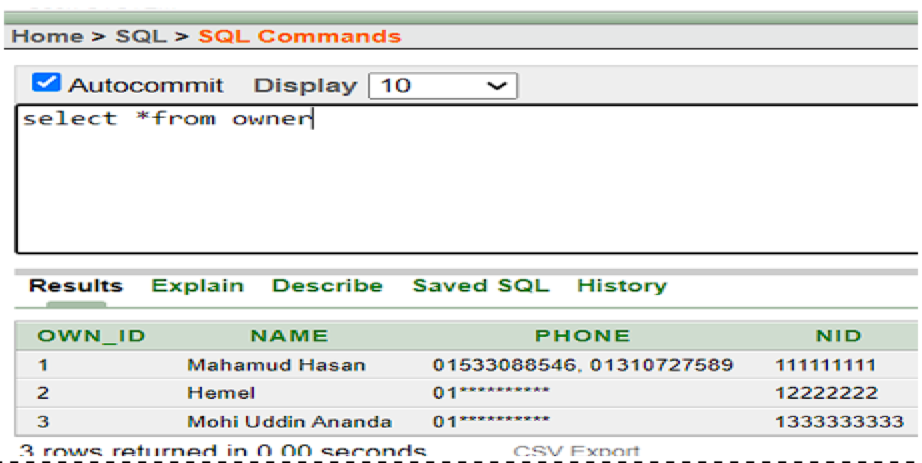
own_id int NOT NULL PRIMARY KEY,

name varchar(255),

phone varchar(255),

nid int NOT NULL

);


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

```
select *from owner
```

OWN_ID	NAME	PHONE	NID
1	Mahamud Hasan	01533088546, 01310727589	111111111
2	Hemel	01***********	12222222
3	Mohi Uddin Ananda	01***********	1333333333

```
3 rows returned in 0.00 seconds
```

create table manager(

mgr_id int NOT NULL PRIMARY KEY,

name varchar(50),

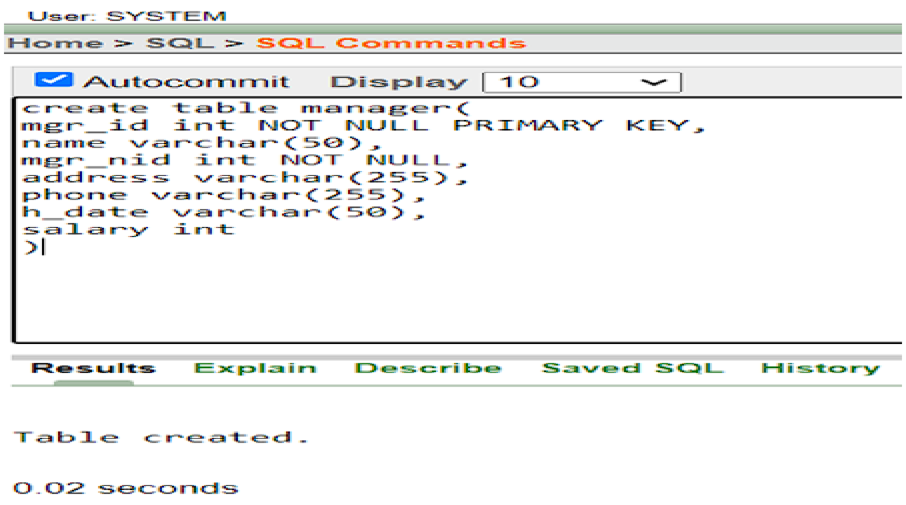
mgr_nid int NOT NULL,

address varchar(255), phone varchar(255),

h_date varchar(50),

salary int

)


User: SYSTEM
Home > SQL > SQL Commands
Autocommit Display 10
Results Explain Describe Saved SQL History

```
create table manager(
mgr_id int NOT NULL PRIMARY KEY,
name varchar(50),
mgr_nid int NOT NULL,
address varchar(255),
phone varchar(255),
h_date varchar(50),
salary int
)
```

```
Table created.

0.02 seconds
```

create table employee(

emp_id int NOT NULL PRIMARY KEY,

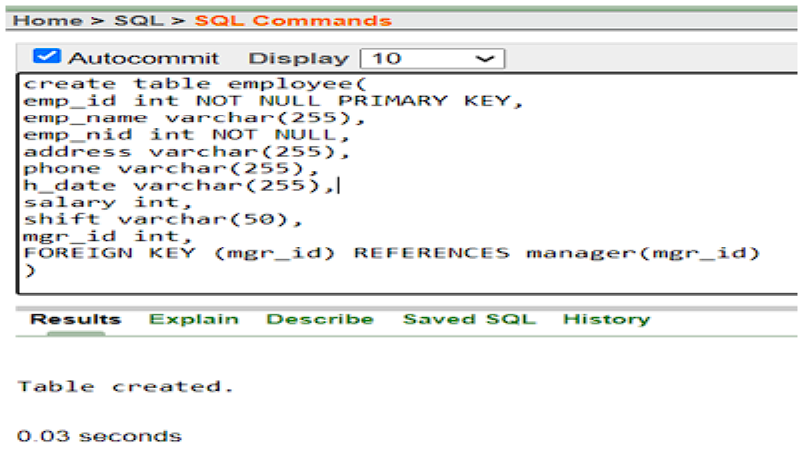
emp_name varchar(255),

emp_nid int NOT NULL,

address varchar(255),phone varchar(255), h_date varchar(255), salary int, shift varchar(50), mgr_id int,

FOREIGN KEY (mgr_id) REFERENCES manager(mgr_id)

)



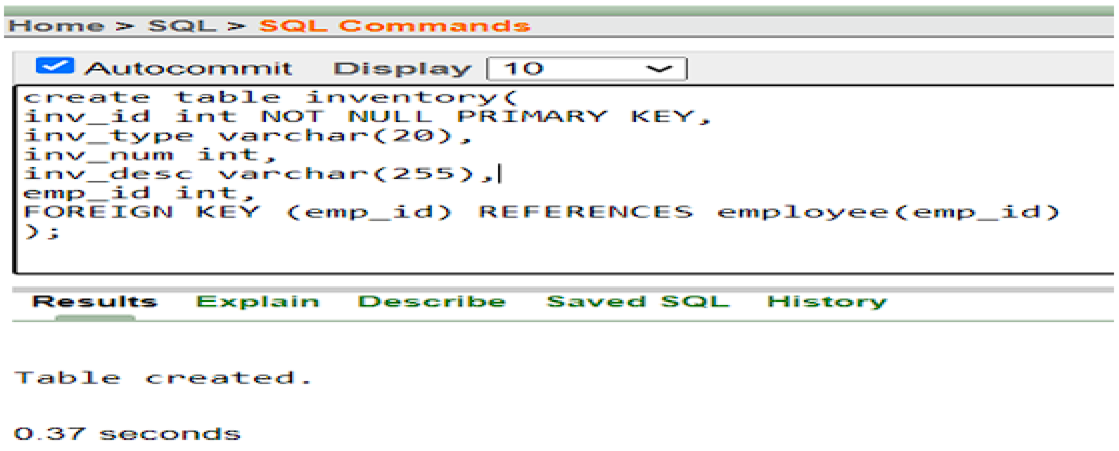
```
create table employee(
emp_id int NOT NULL PRIMARY KEY,
emp_name varchar(255),
emp_nid int NOT NULL,
address varchar(255),
phone varchar(255),
h_date varchar(255),
salary int,
shift varchar(50),
mgr_id int,
FOREIGN KEY (mgr_id) REFERENCES manager(mgr_id)
)
```

```
Table created.

0.03 seconds
```

create table inventory(

inv_id int NOT NULL PRIMARY KEY, inv_type varchar(20), inv_num int, inv_desc varchar(255), emp_id int, FOREIGN KEY (emp_id) REFERENCES employee(emp_id)

);



```
create table inventory(
inv_id int NOT NULL PRIMARY KEY,
inv_type varchar(20),
inv_num int,
inv_desc varchar(255),
emp_id int,
FOREIGN KEY (emp_id) REFERENCES employee(emp_id)
);
```

```
Table created.

0.37 seconds
```

create table STOCKS(

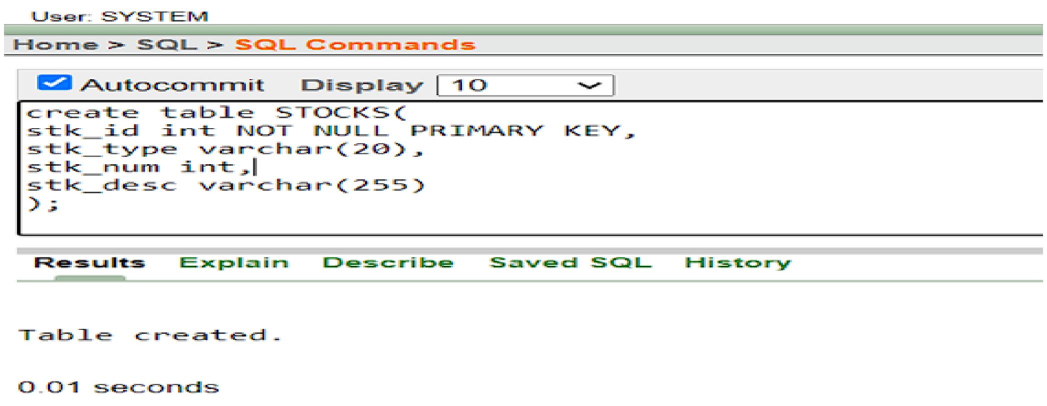
stk_id int NOT NULL PRIMARY KEY,

stk_type varchar(20),

stk_num int,

stk_desc varchar(255)

);


User: SYSTEM
Home > SQL > SQL Commands

```
create table STOCKS(
stk_id int NOT NULL PRIMARY KEY,
stk_type varchar(20),
stk_num int,
stk_desc varchar(255)
);
```

```
Table created.

0.01 seconds
```

create table brands(

brd_id int NOT NULL PRIMARY KEY,

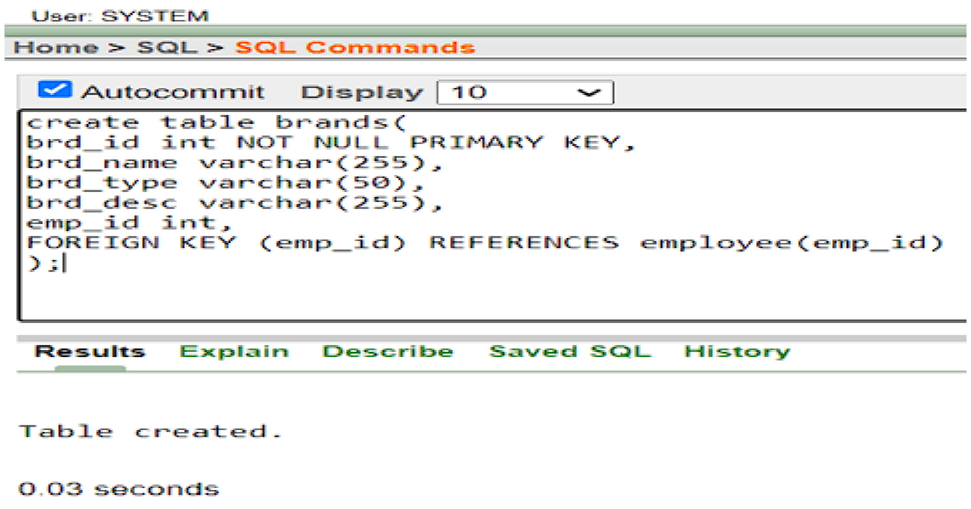
brd_name varchar(255),

brd_type varchar(50),

brd_desc varchar(255),

emp_id int, FOREIGN KEY (emp_id) REFERENCES employee(emp_id)

);


User: SYSTEM
Home > SQL > SQL Commands

```
create table brands(
brd_id int NOT NULL PRIMARY KEY,
brd_name varchar(255),
brd_type varchar(50),
brd_desc varchar(255),
emp_id int,
FOREIGN KEY (emp_id) REFERENCES employee(emp_id)
);
```

```
Table created.

0.03 seconds
```

create table sales(

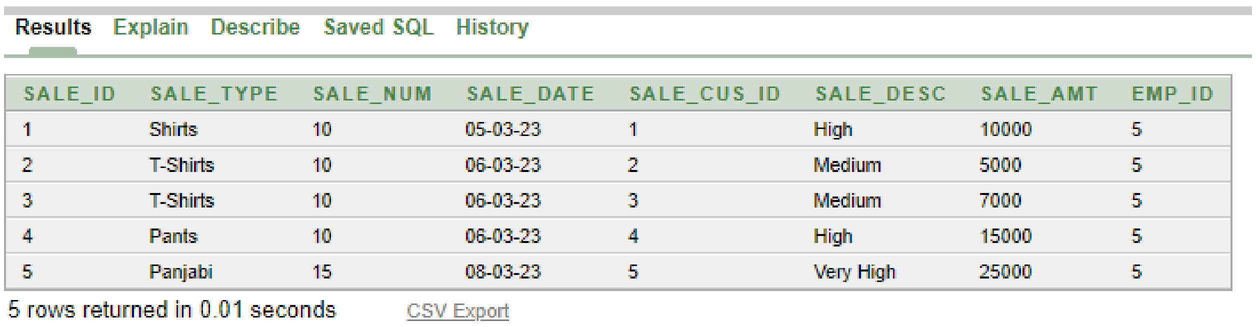
sale_id int NOT NULL PRIMARY KEY,

sale_type varchar(255),

sale_num int,

sale_date varchar(50), sale_cus_id int , sale_desc varchar(255), sale_amt int, emp_id int Not Null, FOREIGN KEY (emp_id) REFERENCES employee(emp_id)

);



SALE_ID	SALE_TYPE	SALE_NUM	SALE_DATE	SALE_CUS_ID	SALE_DESC	SALE_AMT	EMP_ID
1	Shirts	10	05-03-23	1	High	10000	5
2	T-Shirts	10	06-03-23	2	Medium	5000	5
3	T-Shirts	10	06-03-23	3	Medium	7000	5
4	Pants	10	06-03-23	4	High	15000	5
5	Panjabi	15	08-03-23	5	Very High	25000	5

5 rows returned in 0.01 seconds CSV Export

USER PRIVILEGESE:

create user Manager identified by tiger

CREATE ROLE Manage;

GRANT ALL PRIVILEGES TO Manage;

Grant Manage to Manager;

create user employee identified by tiger

Grant unlimited tablespace to employee;

CREATE ROLE employment;

GRANT employment to employee ;

GRANT SELECT, INSERT, UPDATE, DELETE ON system.inventory TO employment;

GRANT SELECT, INSERT, UPDATE, DELETE ON system.stocks TO employment;

GRANT SELECT, INSERT, UPDATE, DELETE ON system.brands TO employment;

GRANT SELECT, INSERT, UPDATE, DELETE ON system.sales TO employment;

GRANT CREATE TABLE TO employment;

GRANT CREATE SEQUENCE TO employment;

GRANT CREATE SYNONYM TO employment;

GRANT CREATE view TO employment;

Creating Index:

CREATE INDEX idx1 ON company (location);

CREATE INDEX idx2 ON owner (name);

```
CREATE INDEX idx3 ON manager (name);
CREATE INDEX idx4 ON employee(emp_name);
CREATE INDEX idx5 ON inventory(inv_type);
CREATE INDEX idx6 ON stocks (stk_num);
CREATE INDEX idx7 ON brands (brd_name);
CREATE INDEX idx8 ON sales (sale_num);
```

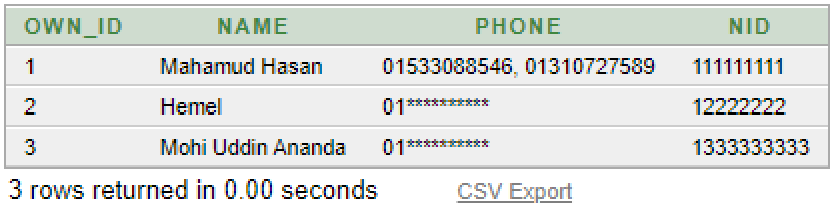
Sequence:

```
CREATE SEQUENCE employee_sq
  INCREMENT BY 1
  START WITH 1
  MAXVALUE 100
  NOCACHE
  NOCYCLE;
describe employee_sq

CREATE SEQUENCE sales_sq
  INCREMENT BY 1
  START WITH 1
  MAXVALUE 100
  NOCACHE
  NOCYCLE;
```

10. Data Insertion

```
insert into company(trade_lc_no ,open_date ,location)

values (0101,'22-02-23','Puran Dhaka')
```

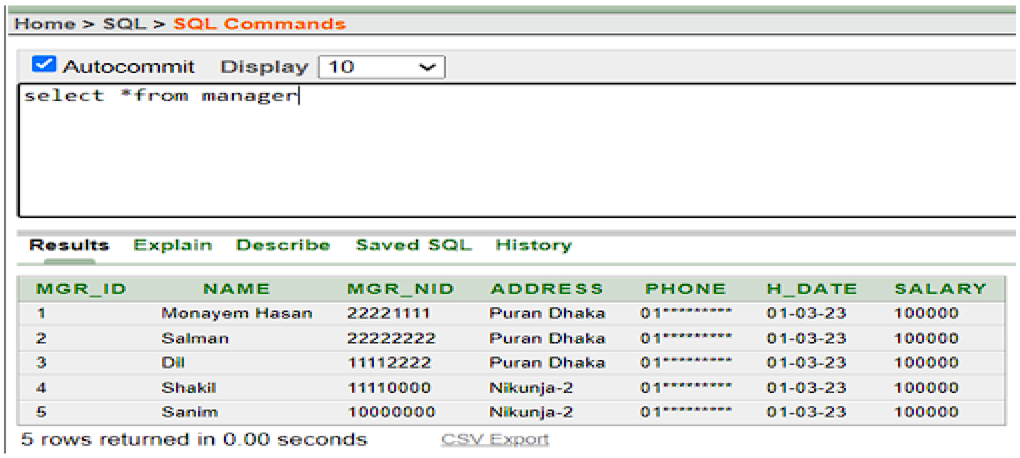
```
insert into owner(own_id,name,phone,nid)

values(01,'Mahamud Hasan','01533088546, 01310727589',111111111)

insert into owner(own_id,name,phone,nid)

values(02,'Hemel','01**********',12222222)

insert into owner(own_id,name,phone,nid)

values(03,'Mohi Uddin Ananda','01**********',1333333333)
```



OWN_ID	NAME	PHONE	NID
1	Mahamud Hasan	01533088546, 01310727589	111111111
2	Hemel	01**********	12222222
3	Mohi Uddin Ananda	01**********	1333333333

3 rows returned in 0.00 seconds CSV Export

```
insert into manager(mgr_id,name,mgr_nid,address,phone,h_date,salary)

values(01,'Monayem Hasan',22221111,'Puran Dhaka','01*********','01-03-23',100000)

insert into manager(mgr_id,name,mgr_nid,address,phone,h_date,salary)

values(02,'Salman',22222222,'Puran Dhaka','01*********','01-03-23',100000)

insert into manager(mgr_id,name,mgr_nid,address,phone,h_date,salary)

values(03,'Dil',11112222,'Puran Dhaka','01*********','01-03-23',100000)

insert into manager(mgr_id,name,mgr_nid,address,phone,h_date,salary)

values(04,'Shakil',11110000,'Nikunja-2','01*********','01-03-23',100000)

insert into manager(mgr_id,name,mgr_nid,address,phone,h_date,salary)

values(05,'Sanim',10000000,'Nikunja-2','01*********','01-03-23',100000)
```


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

```
select *from manager
```

MGR_ID	NAME	MGR_NID	ADDRESS	PHONE	H_DATE	SALARY
1	Monayem Hasan	22221111	Puran Dhaka	01*********	01-03-23	100000
2	Salman	22222222	Puran Dhaka	01*********	01-03-23	100000
3	Dil	11112222	Puran Dhaka	01*********	01-03-23	100000
4	Shakil	11110000	Nikunja-2	01*********	01-03-23	100000
5	Sanim	10000000	Nikunja-2	01*********	01-03-23	100000

5 rows returned in 0.00 seconds CSV Export

insert into employee(emp_id ,emp_name,emp_nid,address,phone,h_date,salary,shift,mgr_id)

values(01,'Pantho',99999999,'Nikunja-2','01*********','02-03-23',20000,'Morning',01)

insert into employee(emp_id ,emp_name,emp_nid,address,phone,h_date,salary,shift,mgr_id)

values(02,'Hasib',99999998,'Nikunja-2','01*********','02-03-23',20000,'Morning',02)

insert into employee(emp_id ,emp_name,emp_nid,address,phone,h_date,salary,shift,mgr_id)

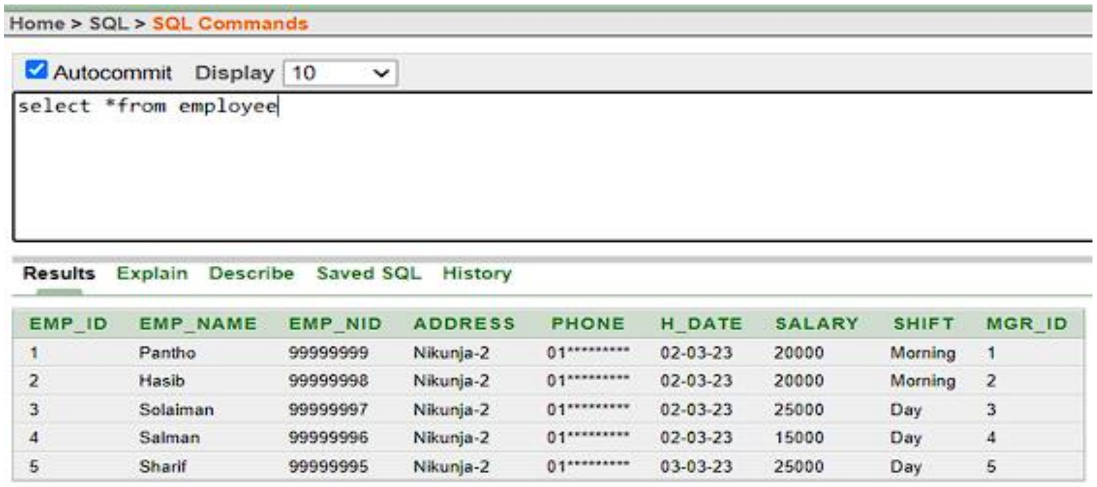
values(03,'Solaiman',99999997,'Nikunja-2','01*********','02-03-23',25000,'Day',03)

insert into employee(emp_id ,emp_name,emp_nid,address,phone,h_date,salary,shift,mgr_id)

values(04,'Salman',99999996,'Nikunja-2','01*********','02-03-23',15000,'Day',04)

insert into employee(emp_id ,emp_name,emp_nid,address,phone,h_date,salary,shift,mgr_id)

values(05,'Sharif',99999995,'Nikunja-2','01*********','03-03-23',35000,'Day',05)


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

```
select *from employee
```

EMP_ID	EMP_NAME	EMP_NID	ADDRESS	PHONE	H_DATE	SALARY	SHIFT	MGR_ID
1	Pantho	99999999	Nikunja-2	01*********	02-03-23	20000	Morning	1
2	Hasib	99999998	Nikunja-2	01*********	02-03-23	20000	Morning	2
3	Solaiman	99999997	Nikunja-2	01*********	02-03-23	25000	Day	3
4	Salman	99999996	Nikunja-2	01*********	02-03-23	15000	Day	4
5	Sharif	99999995	Nikunja-2	01*********	03-03-23	25000	Day	5

insert into inventory(inv_id,inv_type ,inv_num ,inv_desc,emp_id)

values(01,'Shirts',10,'All Good!',01)

insert into inventory(inv_id,inv_type ,inv_num ,inv_desc,emp_id)

values(02,'Pants',10,'All Good!',01)

insert into inventory(inv_id,inv_type ,inv_num ,inv_desc,emp_id)

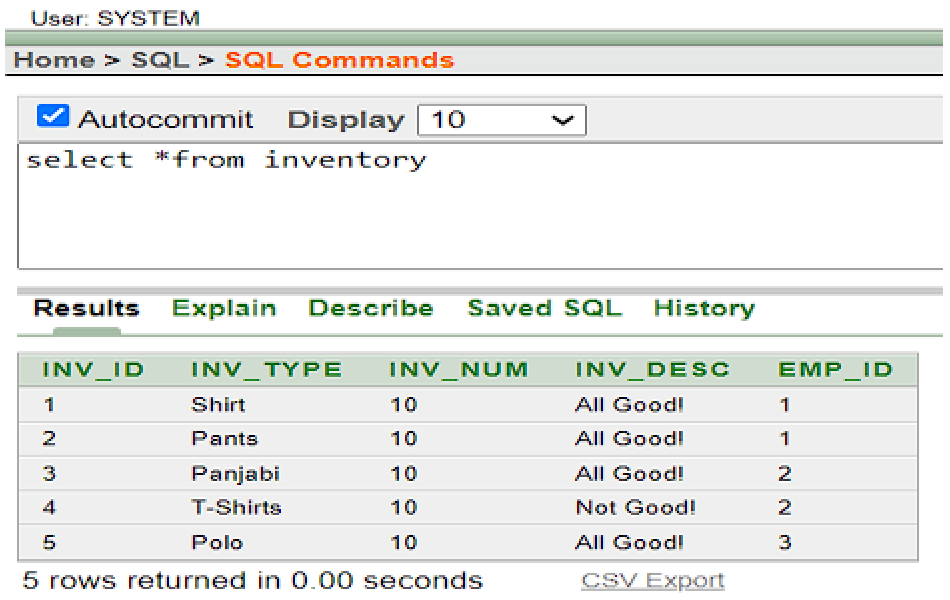
values(03,'Panjabi',10,'All Good!',02)

insert into inventory(inv_id,inv_type ,inv_num ,inv_desc,emp_id)

values(04,'T-Shirts',10,'Not Good!',02)

insert into inventory(inv_id,inv_type ,inv_num ,inv_desc,emp_id)

values(05,'Polo',10,'All Good!',03)


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

```
select *from inventory
```

INV_ID	INV_TYPE	INV_NUM	INV_DESC	EMP_ID
1	Shirt	10	All Good!	1
2	Pants	10	All Good!	1
3	Panjabi	10	All Good!	2
4	T-Shirts	10	Not Good!	2
5	Polo	10	All Good!	3

5 rows returned in 0.00 seconds

insert into STOCKS(stk_id ,stk_type ,stk_num ,stk_desc)

VALUES (01,'Shirts','10','Good')

insert into STOCKS(stk_id ,stk_type ,stk_num ,stk_desc)

VALUES (02,'Shirts','10','Good')

insert into STOCKS(stk_id ,stk_type ,stk_num ,stk_desc)

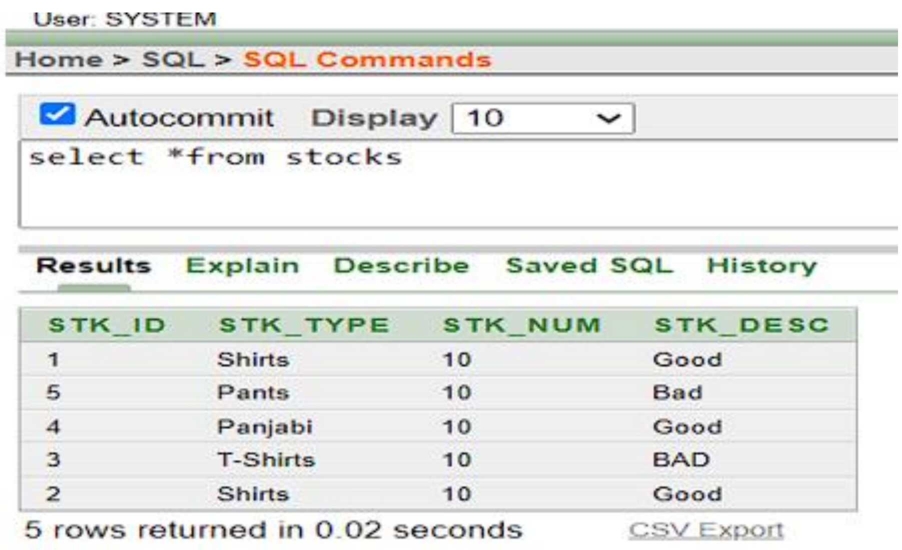
VALUES (03,'T-Shirts','10','BAD')

insert into STOCKS(stk_id ,stk_type ,stk_num ,stk_desc)

VALUES (04,'Panjabi','10','Good')

insert into STOCKS(stk_id ,stk_type ,stk_num ,stk_desc)

VALUES (05,'Pants','10','Bad')


User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

```
select *from stocks
```

Results Explain Describe Saved SQL History

STK_ID	STK_TYPE	STK_NUM	STK_DESC
1	Shirts	10	Good
5	Pants	10	Bad
4	Panjabi	10	Good
3	T-Shirts	10	BAD
2	Shirts	10	Good

5 rows returned in 0.02 seconds CSV Export

insert into brands(brd_id,brd_name,brd_type,brd_desc,emp_id)

values(01,'Ecstasy','Mens','Top',04)

insert into brands(brd_id,brd_name,brd_type,brd_desc,emp_id)

values(02,'Sailor','Mens','Top',04)

insert into brands(brd_id,brd_name,brd_type,brd_desc,emp_id)

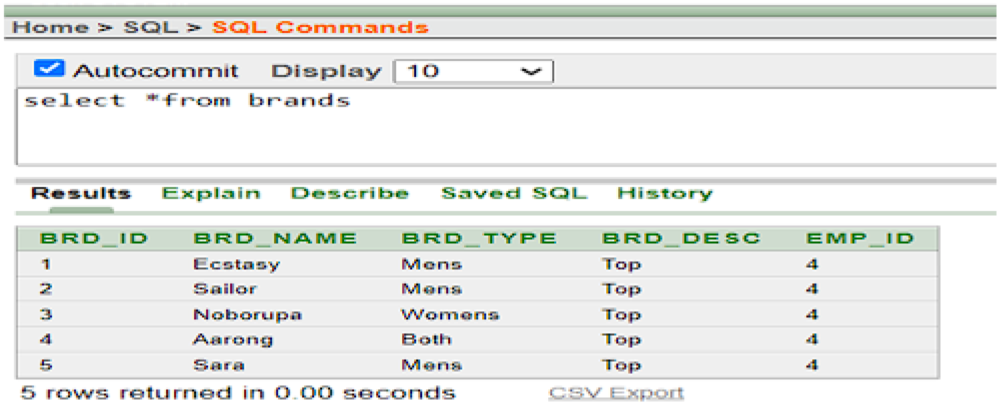
values(03,'Noborupa','Womens','Top',04)

insert into brands(brd_id,brd_name,brd_type,brd_desc,emp_id)

values(04,'Aarong','Both','Top',04)

insert into brands(brd_id,brd_name,brd_type,brd_desc,emp_id)

values(05,'Sara','Mens','Top',04)


Home > SQL > SQL Commands

Autocommit Display 10

```
select *from brands
```

Results Explain Describe Saved SQL History

BRD_ID	BRD_NAME	BRD_TYPE	BRD_DESC	EMP_ID
1	Ecstasy	Mens	Top	4
2	Sailor	Mens	Top	4
3	Noborupa	Womens	Top	4
4	Aarong	Both	Top	4
5	Sara	Mens	Top	4

5 rows returned in 0.00 seconds CSV Export

insert into sales(sale_id,sale_type,sale_num,sale_date,sale_cus_id,sale_desc,sale_amt,emp_id) values (01,'Shirts',10,'05-03-23',01,'High',10000,05)

insert into sales(sale_id,sale_type,sale_num,sale_date,sale_cus_id,sale_desc,sale_amt,emp_id)

values (02,'T-Shirts',10,'06-03-23',02,'Medium',5000,05)

insert into sales(sale_id,sale_type,sale_num,sale_date,sale_cus_id,sale_desc,sale_amt,emp_id)

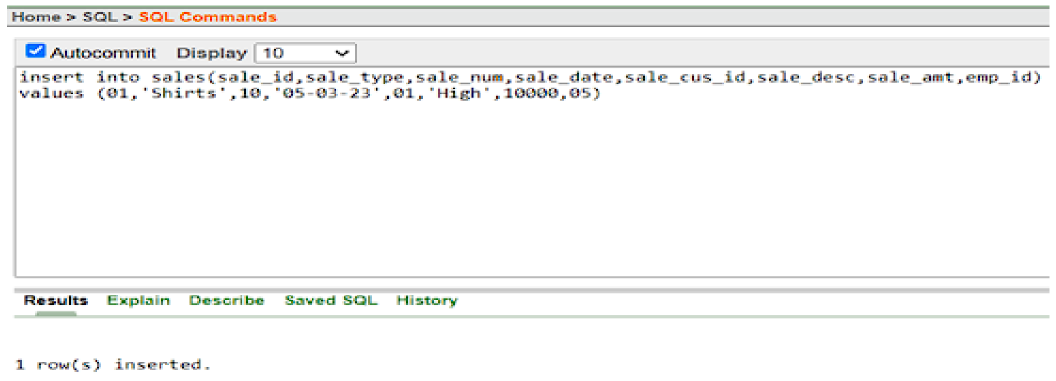
values (03,'T-Shirts',10,'06-03-23',03,'Medium',7000,05)

insert into sales(sale_id,sale_type,sale_num,sale_date,sale_cus_id,sale_desc,sale_amt,emp_id)

values (04,'Pants',10,'06-03-23',04,'High',15000,05)

insert into sales(sale_id,sale_type,sale_num,sale_date,sale_cus_id,sale_desc,sale_amt,emp_id)

values (05,'Panjabi',15,'08-03-23',05,'Very High',25000,05)



```
insert into sales(sale_id,sale_type,sale_num,sale_date,sale_cus_id,sale_desc,sale_amt,emp_id)
values (01,'Shirts',10,'05-03-23',01,'High',10000,05)
```

```
1 row(s) inserted.
```

10. Query Writing

Subquery:

1. Display the employee names who joined before Sharif.

select emp_name from employee
where h_date < (
select h_date from employee
where emp_name = 'Sharif');

2. Display the employee names who earn less than employee Sharif.

select emp_name, salary from employee
where salary < all (

select min(salary) from employee
where emp_name = 'Sharif');

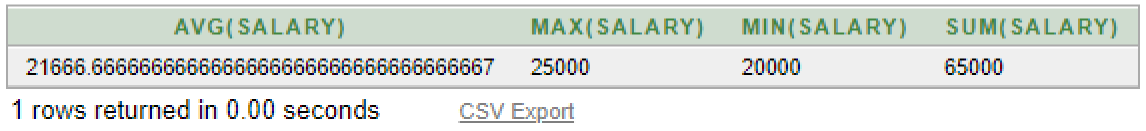
3. Display emp id, customer id, and amount who are managed by an employee

```
SELECT emp_id, sale_cus_id,sale_amt
FROM sales
WHERE emp_id IN (
  SELECT emp_id
  FROM employee
);
```

<u>Single row function</u>

1. Display the avg, min, max, and sum of salary, those whose salarystart with the numeric character 2

SELECT AVG(salary), MAX(salary), MIN(salary), SUM(salary) FROM employee WHERE salary LIKE '2%';

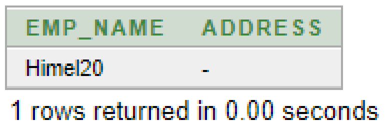


AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
21666.6666666666666666666666666666666667	25000	20000	65000

1 rows returned in 0.00 seconds CSV Export

2. Display the employee name who has no data of their address

SELECT emp_name , addess FROM employee WHERE address IS NULL; (edited)

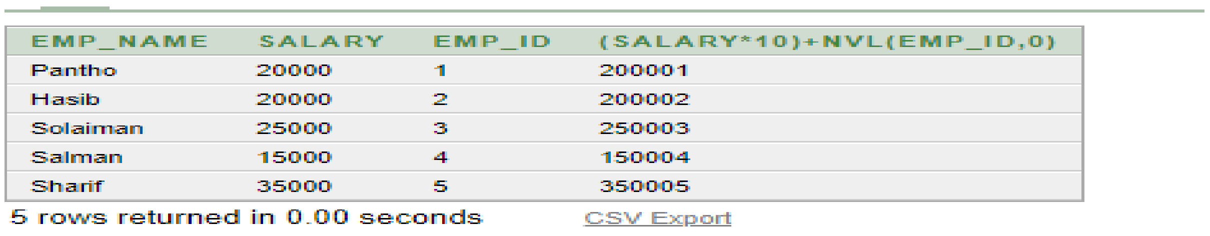


EMP_NAME	ADDRESS
Himel20	-

1 rows returned in 0.00 seconds

3. What is the result of multiplying the salary of each employee by 10, adding their employee ID (if available) to the result, and what are the corresponding employee name, salary, and employee ID in the employee table?

SELECT emp_name , salary, emp_id , (salary*10)+NVL(emp_id ,0) FROM employee;



EMP_NAME	SALARY	EMP_ID	(SALARY*10)+NVL(EMP_ID,0)
Pantho	20000	1	200001
Hasib	20000	2	200002
Solaiman	25000	3	250003
Salman	15000	4	150004
Sharif	35000	5	350005

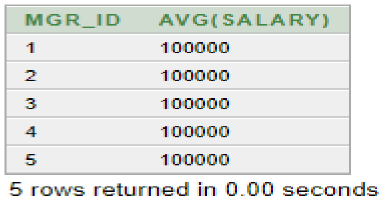
5 rows returned in 0.00 seconds CSV Export

<u>Group by function</u>

1. What is the average salary of employees under each manager, grouped by their manager ID, in the manager table?

SELECT mgr_id , AVG(salary)

FROM manager GROUP BY mgr_id



MGR_ID	AVG(SALARY)
1	100000
2	100000
3	100000
4	100000
5	100000

5 rows returned in 0.00 seconds

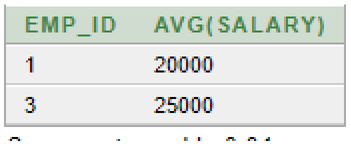
2. What is the average salary of employees with IDs 1, 3, and 6, who have a maximum salary exceeding $5000, and what are their corresponding employee IDs in the employee table?

SELECT emp_id , AVG(salary)

FROM employee WHERE emp_id IN (1, 3, 6)

GROUP BY emp_id

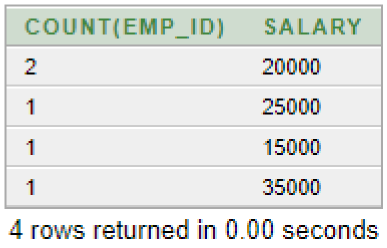
HAVING MAX(salary) > 5000



EMP_ID	AVG(SALARY)
1	20000
3	25000

3. How many employees share each salary amount, and what is the salary value, sorted in descending order based on the count of employees with that salary in the employee table?

SELECT COUNT(emp_id), salary FROM employee GROUP BY salary ORDER BY COUNT(emp_id) DESC;



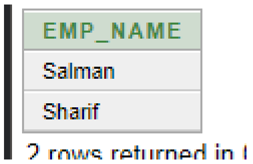
COUNT(EMP_ID)	SALARY
2	20000
1	25000
1	15000
1	35000

4 rows returned in 0.00 seconds

Joining

1. Display the name of all the employees who work in Nikunja-2.

SELECT emp_name FROM employee WHERE mgr_id IN (SELECT mgr_id FROM manager WHERE ADDRESS = 'Nikunja-2');



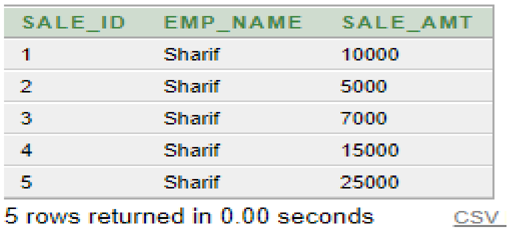
EMP_NAME
Salman
Sharif

2 rows returned in

2. Display the sale id, sale amount.

SELECT sales.sale_id, employee.emp_name, sales.sale_amt

FROM sales INNER JOIN employee ON sales.emp_id=employee.emp_id;

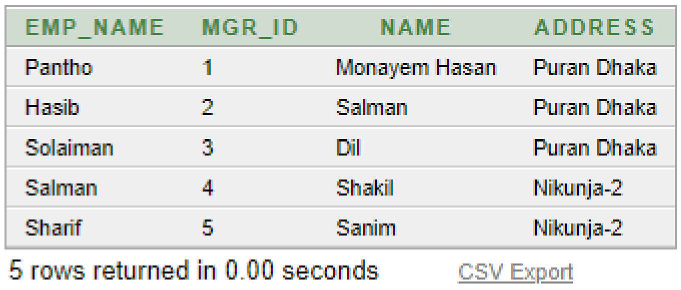


SALE_ID	EMP_NAME	SALE_AMT
1	Sharif	10000
2	Sharif	5000
3	Sharif	7000
4	Sharif	15000
5	Sharif	25000

5 rows returned in 0.00 seconds CSV

3. Write a query to display the name, department number, department name and department location for all employees.

SELECT e.emp_name, e.mgr_id, d.name, d.address FROM employee e, manager d WHERE e.emp_id = d.mgr_id;



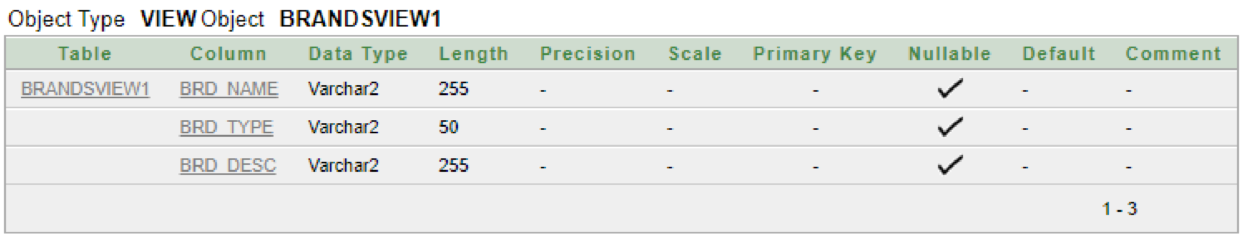
EMP_NAME	MGR_ID	NAME	ADDRESS
Pantho	1	Monayem Hasan	Puran Dhaka
Hasib	2	Salman	Puran Dhaka
Solaiman	3	Dil	Puran Dhaka
Salman	4	Shakil	Nikunja-2
Sharif	5	Sanim	Nikunja-2

5 rows returned in 0.00 seconds CSV Export

<u>View:</u>

1. What is the name, type, and description of the brand with an ID of 1, as retrieved from the "brandsView1" view?

```
CREATE VIEW brandsView1

   AS SELECT brd_name, brd_type, brd_desc

   FROM brands WHERE brd_id = 1;

describe brandsView1
```



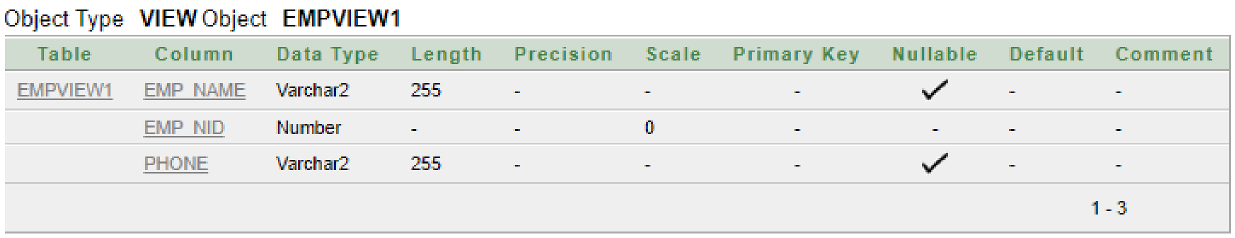
Object Type **VIEW** Object **BRANDSVIEW1**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BRANDSVIEW1	BRD_NAME	Varchar2	255	-	-	-	✓	-	-
	BRD_TYPE	Varchar2	50	-	-	-	✓	-	-
	BRD_DESC	Varchar2	255	-	-	-	✓	-	-

1 - 3

2. What is the name, national identification number, and phone number of the employee with an ID of 1, as retrieved from the "empView1" view?

```
CREATE VIEW empView1

   AS SELECT emp_name, emp_nid, phone

   FROM Employee  WHERE Emp_id = 1;

describe empView1
```



Object Type **VIEW** Object **EMPVIEW1**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
EMPVIEW1	EMP_NAME	Varchar2	255	-	-	-	✓	-	-
	EMP_NID	Number	-	-	0	-	-	-	-
	PHONE	Varchar2	255	-	-	-	✓	-	-

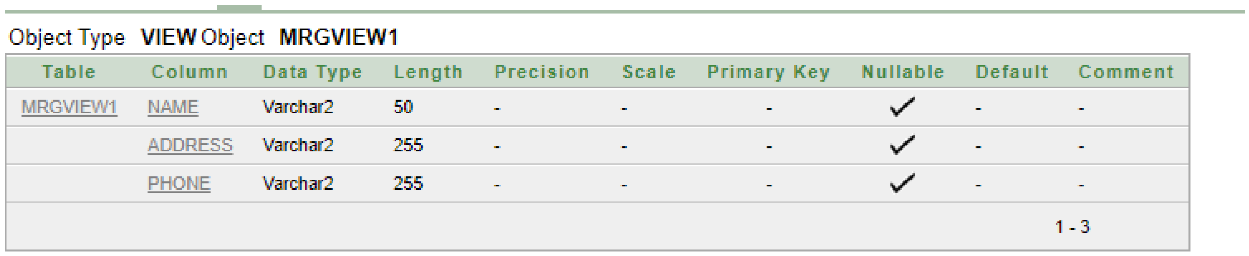
1 - 3

3. What are the name, address, and phone number of the manager who has a salary of $100,000, as retrieved from the "MrgView1" view?

```
CREATE VIEW MrgView1

   AS SELECT name, address, phone
```

```
  FROM Manager WHERE salary =100000;
```

describe MrgView1



Object Type VIEW Object MRGVIEW1

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MRGVIEW1	NAME	Varchar2	50	-	-	-	✓	-	-
	ADDRESS	Varchar2	255	-	-	-	✓	-	-
	PHONE	Varchar2	255	-	-	-	✓	-	-

1 - 3

Synonyms:

1. Create a synonym for sale id

Create SYNONYM sid for sales.sale_id

2. Create a synonym for sale amt

Create SYNONYM amount for sales.sale_amt

3. Create a synonym for emp name

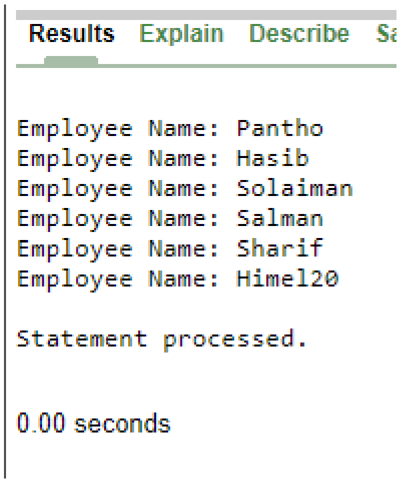
Create SYNONYM em_name for employee.emp_name

PL/SQL

FUNCTIONS:

1. Retrieves the name of an employee based on the given employee ID.

```
CREATE OR REPLACE FUNCTION GetEmployeeName(emp_id IN INT)
RETURN VARCHAR2
IS emp_name VARCHAR2(255);
BEGIN
  FOR emp_rec IN (SELECT emp_name FROM employee WHERE emp_id = emp_id)
  LOOP
    emp_name := emp_rec.emp_name;
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_name);
  END LOOP;
  RETURN emp_name;
END;
DECLARE
  emp_name VARCHAR2(255);
BEGIN
  emp_name := GetEmployeeName(1); -- Replace 1 with the desired employee ID
END;
```

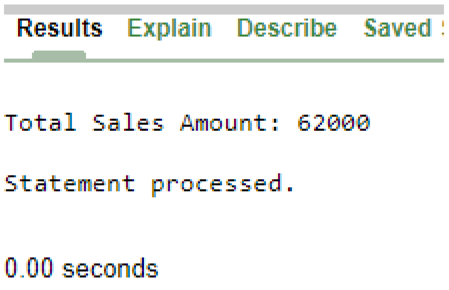


```
Employee Name: Pantho
Employee Name: Hasib
Employee Name: Solaiman
Employee Name: Salman
Employee Name: Sharif
Employee Name: Himel20

Statement processed.


0.00 seconds
```

2. Calculates the total sales amount for a given employee.

PL/SQL Code:

```
CREATE OR REPLACE FUNCTION GetTotalSalesAmount(emp_id IN INT)
RETURN NUMBER
IS  total_amount NUMBER;
BEGIN
  SELECT SUM(sale_amt) INTO total_amount
  FROM sales WHERE emp_id = emp_id;
  DBMS_OUTPUT.PUT_LINE('Total Sales Amount: ' || total_amount);
  RETURN total_amount;
END;
DECLARE
  total_amount NUMBER;
BEGIN
  total_amount := GetTotalSalesAmount(5); -- Replace 5 with the desired employee ID
END;
```



```
Total Sales Amount: 62000

Statement processed.

0.00 seconds
```

3. Retrieves the details (name, address, and phone) of a manager based on the given manager ID.

```
CREATE OR REPLACE FUNCTION GetManagerDetails(mgr_id IN INT)
RETURN VARCHAR2
IS manager_name VARCHAR2(255);
BEGIN
  -- Define a cursor to fetch the results
  FOR manager_rec IN (SELECT name FROM manager WHERE mgr_id = mgr_id)
  LOOP
    manager_name := manager_rec.name;
    DBMS_OUTPUT.PUT_LINE('Manager Name: ' || manager_name);
  END LOOP;
RETURN manager_name;
END;
```

DECLARE

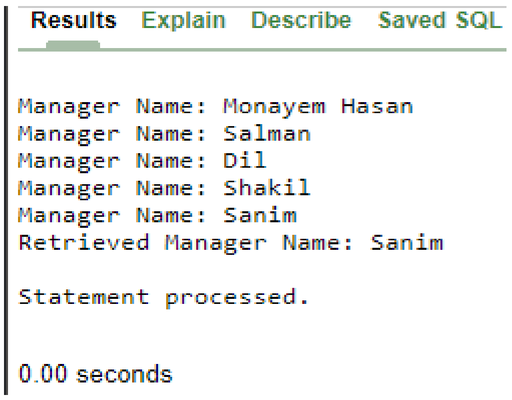
  manager_name VARCHAR2(255);

BEGIN

  manager_name := GetManagerDetails(1); -- Replace 1 with the desired manager ID

  DBMS_OUTPUT.PUT_LINE('Retrieved Manager Name: '|| manager_name);

END;



```
Manager Name: Monayem Hasan
Manager Name: Salman
Manager Name: Dil
Manager Name: Shakil
Manager Name: Sanim
Retrieved Manager Name: Sanim

Statement processed.


0.00 seconds
```

Procedure:

1. Procedure to Retrieve Employee Name and Manager Name::

CREATE OR REPLACE FUNCTION GetManagerDetails(mgr_id IN INT)

RETURN VARCHAR2 IS manager_name VARCHAR2(255);

BEGIN

  -- Define a cursor to fetch the results

  FOR manager_rec IN (SELECT name FROM manager WHERE mgr_id = mgr_id)

  LOOP

    manager_name := manager_rec.name;

    DBMS_OUTPUT.PUT_LINE('Manager Name: '|| manager_name);

  END LOOP;

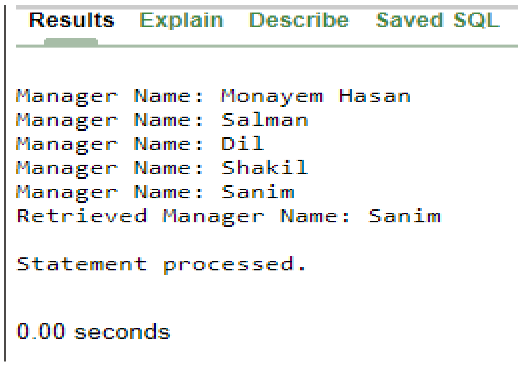
  RETURN manager_name;

END;

DECLARE

  manager_name VARCHAR2(255);

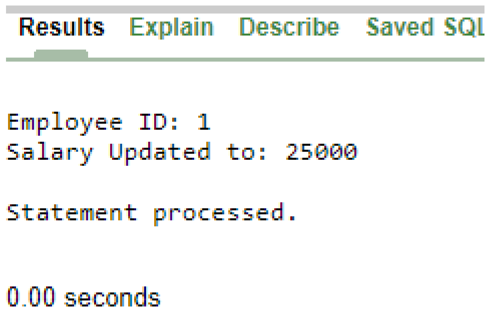
```
BEGIN
  manager_name := GetManagerDetails(1); -- Replace 1 with the desired manager ID
  DBMS_OUTPUT.PUT_LINE('Retrieved Manager Name: '||  manager_name);
END;
```



```
Manager Name: Monayem Hasan
Manager Name: Salman
Manager Name: Dil
Manager Name: Shakil
Manager Name: Sanim
Retrieved Manager Name: Sanim

Statement processed.


0.00 seconds
```

2. Procedure to update Employee Salary:

```
CREATE OR REPLACE PROCEDURE UpdateEmployeeSalary(emp_id IN INT, new_salary IN INT)
IS
BEGIN
  UPDATE employee
  SET salary = new_salary
  WHERE emp_id = emp_id;


  DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_id);
  DBMS_OUTPUT.PUT_LINE('Salary Updated to: '||  new_salary);
END;
```



```
Employee ID: 1
Salary Updated to: 25000

Statement processed.


0.00 seconds
```

3.

CREATE OR REPLACE PROCEDURE UpdateEmployeeSalary(emp_id IN INT, new_salary IN INT)

IS

BEGIN

  UPDATE employee

  SET salary = new_salary

  WHERE emp_id = emp_id;

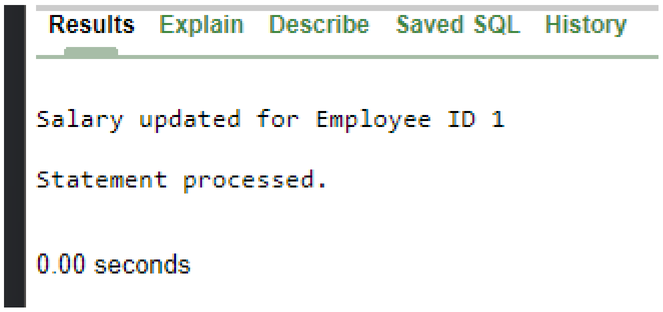
  DBMS_OUTPUT.PUT_LINE('Salary updated for Employee ID ' || emp_id);

END;

BEGIN

  UpdateEmployeeSalary(1, 50000); -- Replace 1 with the desired employee ID and 50000 with the new salary value

END;



```
Salary updated for Employee ID 1

Statement processed.

0.00 seconds
```

<u>RECORD</u>:

1. Record variable from table where prints the employee's name, address, and ID.

declare

emp_rec employee%rowtype;

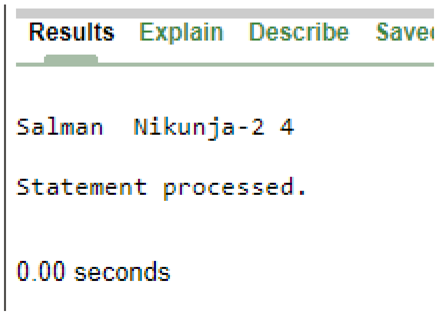
begin

select * into emp_rec from employee

where emp_name='Salman';

dbms_output.put_line(emp_rec .emp_name||' '||emp_rec .address||' '||emp_rec .emp_id);

end



```
Salman  Nikunja-2 4

Statement processed.


0.00 seconds
```

2.

declare

cursor c_emp is

select * from employee where emp_name='Salam';

emp_rec employee%rowtype;

begin

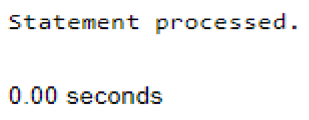
open c_emp;

fetch c_emp into emp_rec;

dbms_output.put_line(emp_rec.emp_id||' '||emp_rec.emp_name||' '||emp_rec.emp_nid||' '||emp_rec.address||' '||emp_rec.phone||' '||emp_rec.h_date||' '||emp_rec.salary||' '||emp_rec.shift);

close c_emp;

end;



```
Statement processed.

0.00 seconds
```

3. retrieves the ID, name, and salary of the first three records from the table

DECLARE

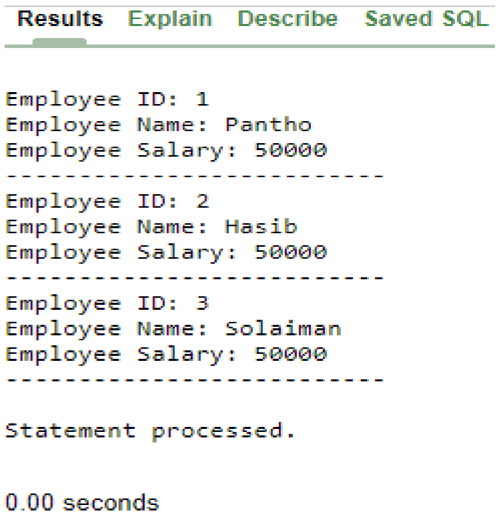
  emp_id   employee.emp_id%TYPE;

  emp_name employee.emp_name%TYPE;

  salary   employee.salary%TYPE;

BEGIN

```
 FOR emp_record IN (
  SELECT emp_id, emp_name, salary
  FROM employee WHERE ROWNUM <= 3 -- Retrieve only three records )
LOOP-- Fetch values into individual variables
  emp_id := emp_record.emp_id;
  emp_name := emp_record.emp_name;
  salary := emp_record.salary;
  -- Print employee details
  DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_id);
  DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_name);
  DBMS_OUTPUT.PUT_LINE('Employee Salary: ' || salary);
  DBMS_OUTPUT.PUT_LINE('--------------------------');
 END LOOP; END;
```

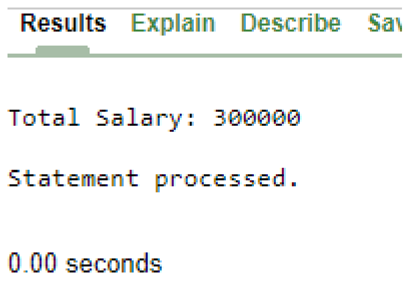


```
Employee ID: 1
Employee Name: Pantho
Employee Salary: 50000
--------------------------
Employee ID: 2
Employee Name: Hasib
Employee Salary: 50000
--------------------------
Employee ID: 3
Employee Name: Solaiman
Employee Salary: 50000
--------------------------

Statement processed.

0.00 seconds
```

<u>CURSOR:</u>

1. Calculating Total Salary

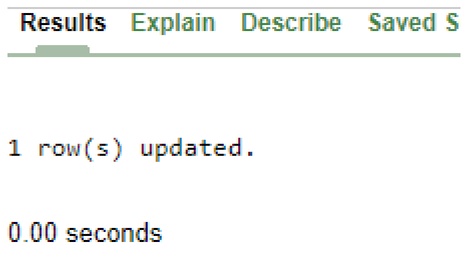
```
DECLARE
 CURSOR emp_cursor IS
  SELECT salary
  FROM employee;
 total_salary NUMBER := 0;
 emp_salary  employee.salary%TYPE;
BEGIN
 OPEN emp_cursor;
 LOOP
  FETCH emp_cursor INTO emp_salary;
  EXIT WHEN emp_cursor%NOTFOUND;
  total_salary := total_salary + emp_salary;
 END LOOP;
 CLOSE emp_cursor;
 -- Print total salary
 DBMS_OUTPUT.PUT_LINE('Total Salary: ' || total_salary);
END;
```



```
Total Salary: 300000

Statement processed.

0.00 seconds
```

2. Updating Employee Salaries

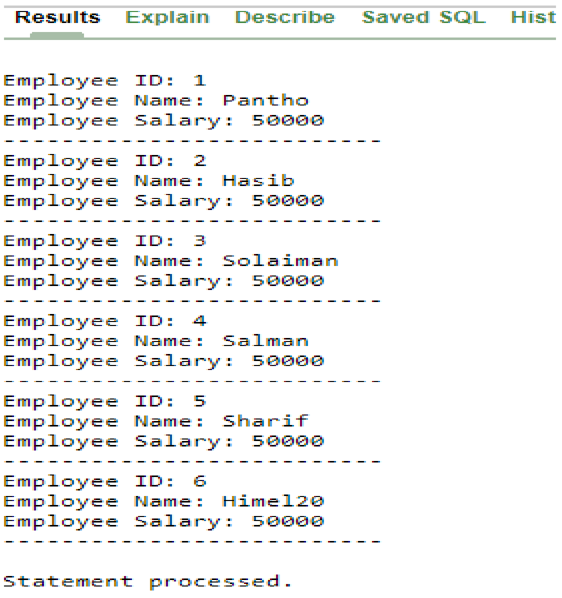
```
DECLARE
  CURSOR emp_cursor IS
    SELECT emp_id, salary
    FROM employee
    WHERE salary < 5000;
  emp_record emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO emp_record;
    EXIT WHEN emp_cursor%NOTFOUND;
    -- Update salary
    emp_record.salary := emp_record.salary * 1.1;
    -- Apply the salary update
    UPDATE employee
    SET salary = emp_record.salary
    WHERE emp_id = emp_record.emp_id;
  END LOOP;
  CLOSE emp_cursor;
END;
```



```
1 row(s) updated.

0.00 seconds
```

3. Fetching and Printing Employee Details

```
DECLARE
  CURSOR emp_cursor IS
    SELECT emp_id, emp_name, salary FROM employee;
  emp_record emp_cursor%ROWTYPE;
BEGIN
  OPEN emp_cursor;
  LOOP
    FETCH emp_cursor INTO emp_record;
    EXIT WHEN emp_cursor%NOTFOUND;
    -- Print employee details
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.emp_id);
    DBMS_OUTPUT.PUT_LINE('Employee Name: ' || emp_record.emp_name);
    DBMS_OUTPUT.PUT_LINE('Employee Salary: ' || emp_record.salary);
    DBMS_OUTPUT.PUT_LINE('--------------------------');
  END LOOP;
  CLOSE emp_cursor;
END;
```



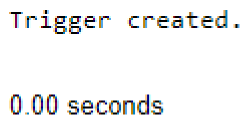
```
Employee ID: 1
Employee Name: Pantho
Employee Salary: 50000
--------------------------
Employee ID: 2
Employee Name: Hasib
Employee Salary: 50000
--------------------------
Employee ID: 3
Employee Name: Solaiman
Employee Salary: 50000
--------------------------
Employee ID: 4
Employee Name: Salman
Employee Salary: 50000
--------------------------
Employee ID: 5
Employee Name: Sharif
Employee Salary: 50000
--------------------------
Employee ID: 6
Employee Name: Himel20
Employee Salary: 50000
--------------------------

Statement processed.
```

TRIGGER:

1. Before Insert Trigger

```
CREATE OR REPLACE TRIGGER before_insert_employee
BEFORE INSERT ON employee
FOR EACH ROW
BEGIN
  IF :NEW.h_date IS NULL THEN
    :NEW.h_date := SYSDATE;
  END IF;
```

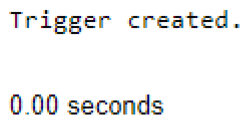
```
END;
```



```
Trigger created.

0.00 seconds
```

2. Update Employee Salary

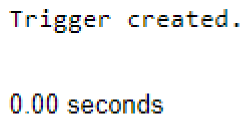
```
CREATE OR REPLACE TRIGGER update_employee_salary
BEFORE INSERT ON employee
FOR EACH ROW
BEGIN
  :NEW.salary := :NEW.salary * 1.1; -- Increase the salary by 10%
END;
```



```
Trigger created.

0.00 seconds
```

3. Prevent Delete on Sales Table

```
CREATE OR REPLACE TRIGGER prevent_delete_sales
BEFORE DELETE ON sales
BEGIN
  RAISE_APPLICATION_ERROR(-20001, 'Deleting records from the sales table is not allowed.');
END;
```



```
Trigger created.

0.00 seconds
```

Package:

1. Sales Package:

CREATE OR REPLACE PACKAGE sales_package IS

PROCEDURE create_sale (sale_id IN NUMBER, sale_type IN VARCHAR2, sale_num IN NUMBER, sale_date IN VARCHAR2, sale_cus_id IN NUMBER, sale_desc IN VARCHAR2, sale_amt IN NUMBER, emp_id IN NUMBER);

PROCEDURE update_sale(sale_id IN NUMBER, new_sale_amt IN NUMBER);

FUNCTION get_sale_details(sale_id IN NUMBER) RETURN VARCHAR2;

END sales_package;

BEGIN

employee_package.insert_new_employee(

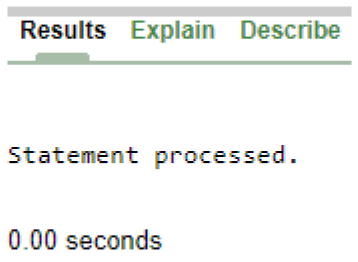
6, 'John Doe', 1234567890, '123 Main St', '123-456-7890',

'2023-05-13', 5000, 'Day', 2

); -- Provide the appropriate values for the new employee

END;

/



```
Statement processed.

0.00 seconds
```

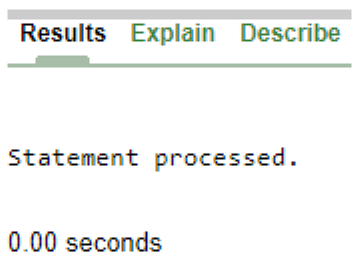
2.

CREATE OR REPLACE PACKAGE employee_package AS

PROCEDURE get_employee_details(emp_id IN NUMBER);

PROCEDURE

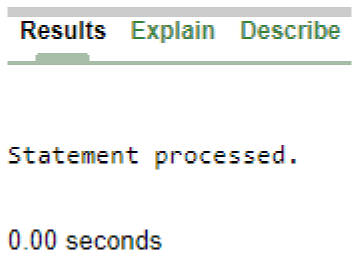
```
insert_new_employee (emp_id IN NUMBER, emp_name IN VARCHAR2, emp_nid IN NUMBER,
address IN VARCHAR2, phone IN VARCHAR2, h_date IN VARCHAR2, salary IN NUMBER,  shift IN
VARCHAR2,mgr_id IN NUMBER
 );
END;
-- Create the package body
CREATE OR REPLACE PACKAGE BODY employee_package AS
 PROCEDURE get_employee_details(emp_id IN NUMBER) IS
  -- Procedure implementation
 BEGIN
  -- Your code here
  NULL;
 END;
 PROCEDURE insert_new_employee(
  emp_id IN NUMBER,
  emp_name IN VARCHAR2,emp_nid IN NUMBER, address IN VARCHAR2, phone IN VARCHAR2,
h_date IN VARCHAR2, salary IN NUMBER, shift IN VARCHAR2,    mgr_id IN NUMBER
 ) IS
  -- Procedure implementation
 BEGIN
  -- Your code here
  NULL;
 END;
END;
BEGIN
 employee_package.get_employee_details(1); -- Provide the desired employee ID
END;
```



```
Statement processed.

0.00 seconds
```

3. Inventory Package:

```
CREATE OR REPLACE PACKAGE inventory_package IS

PROCEDURE add_inventory_item (inv_id IN NUMBER, inv_type IN VARCHAR2, inv_num IN NUMBER, inv_desc IN VARCHAR2, emp_id IN NUMBER);

  PROCEDURE remove_inventory_item(inv_id IN NUMBER);

  FUNCTION check_inventory_item(inv_id IN NUMBER) RETURN VARCHAR2;

END inventory_package;

BEGIN

  sales_package.create_sale(

    6, 'Shirts', 10, '2023-05-13', 1, 'High', 10000, 3

  ); -- Provide the appropriate values for the new sale

END;
```



```
Statement processed.

0.00 seconds
```

12. Relational Algebra

1. Find the name of the staff whose salary is greater than 6000.
2. Find the sale type which sales’s id 1.
3. Find the manager NID from manager table where Manager id number 4.
4. Find the ID of manager named Hasib.

5. Find the brand id of Ecstasy.

Solutions:

1. $\prod_{Name} (\sigma_{salary > 6000} (Manager))$
2. $\prod_{sale\_type} (\sigma_{sale\_id=1} (Sales))$
3. $\prod_{Mgr\_NID} (\sigma_{mrg\_id=4} (Manager))$
4. $\prod_{emp\_ID} (\sigma_{Name='HASIB'} (employee))$
5. $\prod_{brd\_id} (\sigma_{brd\_name='Ecstasy'} (Brands))$

13. Conclusion

In conclusion, the Garment Management System is an essential tool for garment manufacturers and managers. The system is designed to automate and streamline the entire garment manufacturing process, from order placement to inventory management, production tracking, quality control, and delivery management. The application is built using modern technologies such as Oracle 10g database and PHP interface, which provides a robust and secure platform for managing data. It offers a user-friendly interface that makes it easy for users to access and manage data while providing role-based access control that ensures only authorized personnel can access sensitive data.

The system is expected to increase productivity and reduce lead times, thereby improving quality and reducing costs. It also provides real-time reporting capabilities that enable users to make data-driven decisions.

User interface isn’t connected with oracle database. In future, we are planning to make a connection between oracle. Which will give more efficiency of our works.

Overall, the Garment Management System is a comprehensive solution that simplifies and streamlines garment manufacturing operations, improves quality, and reduces costs. It is an essential tool that garment manufacturers and managers can use to stay competitive in the fast-paced world of fashion.