AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

Course Title

INTRODUCTION TO DATABASE (2108)

Semester: Fall 2024-2025

Section: [H]

TITLE

NGO Management System

Supervised By

MD Sajid Bin-Faisal

Submitted By: Group no: 03

Name	ID
Samaun Islam Ikra	23-51584-2
Eshtiak Ferdous Mahi	23-51744-2
Nazmul Hossain	23-51942-2
A S M Shahjalal	23-55671-3

TABLE OF CONTENTS

Introduction

The title of the project is “NGO management system”. To create this database system, we have used DDL and DML to make relational tables and store the information. We also have used normalization to reduce the anomalies and make the database easy to handle.

Case Study / Scenario

In an NGO management system, detailed information is maintained about the CEO, Manager, Workers, Project. People and NGO. The CEO inspects the managers. The CEO can have many managers. But a manager can only One CEO. The database. Stores work under on information about the CEO, for example their name, ID. Hire date, address and number. The NGO is managed by a team of managers. Uniquely identified by their ID, name, salary, address, contact number. Hire-date and which currently working on An NGO can have many managers, but a manager can Work under only one NGO. It's identified by NGO. license, number, fund and address. The address of an NGO typically comprises the city and specific area of its location. The worker reports to the mana- ger. A manager can have many workers. workers can work with only one manager. The workers’ name, unique ID, contact number, salary, address, hire-date, their position will be stored. Workers contribute to the completion of the project. project information such as identify unique ID, Zip-code, name, duration, area, start date, cost and which company is funding will also be stored. Every Project will serve people. A project can be done by many people, but people can work on only One project. The database maintains detailed record of People, including their name, NID, address and contact information. Now, draw on ER Diagram according to the mentioned scenario.

ER Diagram

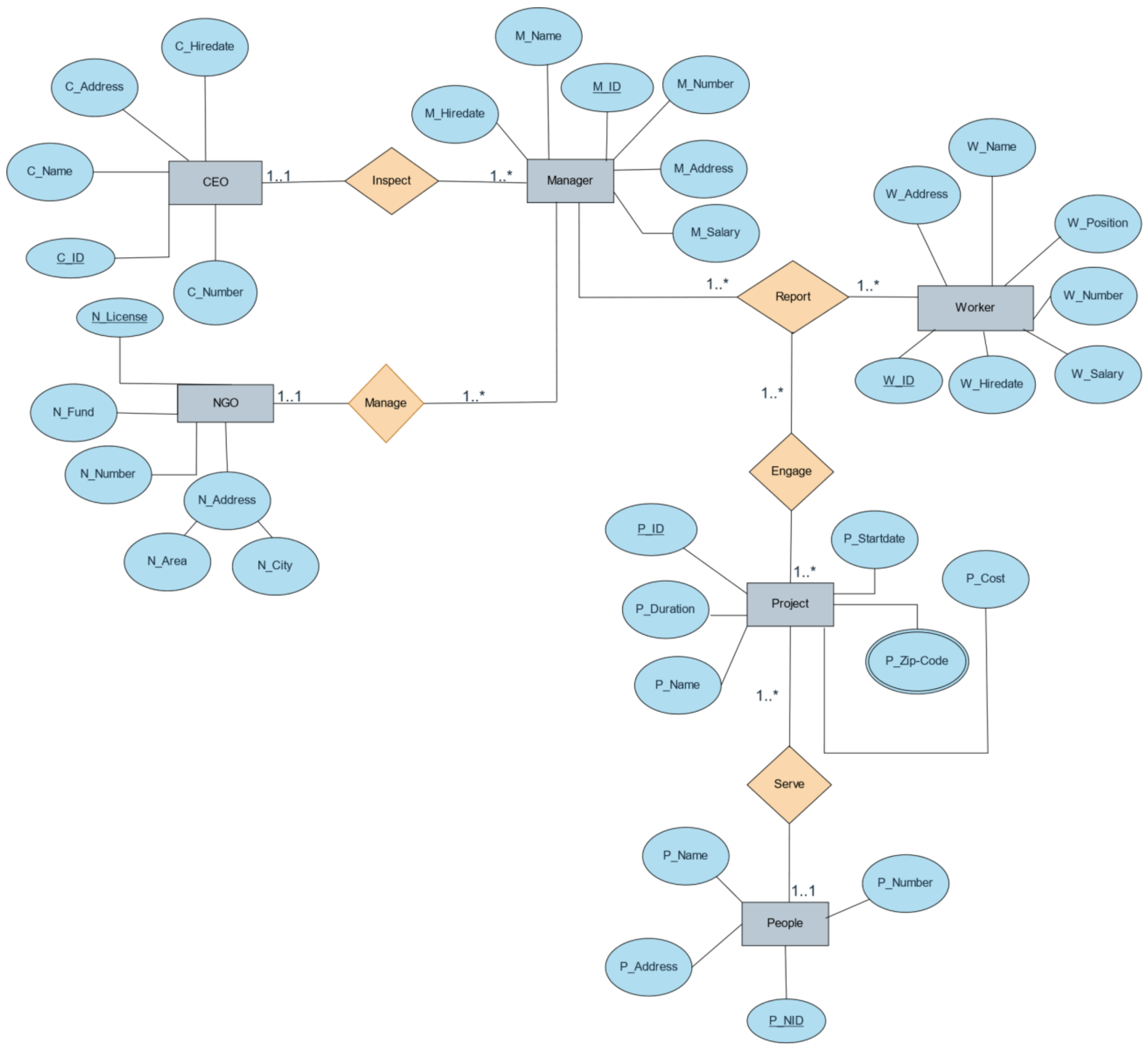


Fig 1: E-R Diagram of NGO Management System

Normalization

CEO-INSPECT-MANAGER (1..*)

UNF:

cid, cname, cadd, chiredate, cnum, mid, mnum, mhiredate, mname, msal,madd

1NF:

cid, cname, cadd, chiredate, cnum, mid, mnum, mhiredate, mname, msal, madd

2NF:

1)cid, cname, cadd, chiredate, cnum
2) mid, mnum, mhiredate, mname, msal, madd,c_id (FK)

3NF:

1)cid, c_name, c_address, c_hiredate, c_number
2)mid, mnum, mhiredate, mname, msal, madd, cid (FK**)**

NGO-MANAGED-MANAGER (1..*)

UNF :

n_license, n_fund, n_number, n_area, n_city, n_address, m_id, m_number, m_hiredate, m_name, m_salary, m_address

1NF :

nlicense, nfund, nnum, narea, ncity, mid, mnum, mhiredate, mname, msal, madd

2NF:

1) nlicense, nfund, nnum, narea

2) mid, mnum, mhiredate, mname, msal, madd ,nlicense (FK)

3NF :

1) nlicense, nfund, nnum, narea

2) mid, mnum, mhiredate, mname, msal, madd ,nlicense (FK)

3)city, narea

PROJECT-SERVE-PEOPLE (1..*)

UNF:

pid, pcost, pname, pduration, pzipcode, pstartdate, penid,penum,pename, padd

1NF:

pid, pcost, pname, pzipcode, pstartdate, pduration, penid, penum, pename, padd

2NF:

1)pid, pcost, pname, pduration, pstartdate, penid (FK)

2)penid, penum, pename

3NF:

1)pid, pcost, pname, pduration, pstartdate, penid (FK)

2)penid, penum, pename

WORKER-ENGAGE-PROJECT (*..*)

UNF:

wname, wid, wnum, wsal, whiredate, wadd, wposition, pname, pid, pcost, pzipcode, pstartdate, pduration

1NF:

wname, wid, wnum, wsal, whiredate, wadd, wposition,pname, pid, pcost, pzipcode, pstartdate, pduration

2NF:

1) wname, wid, wnum, wsal, whiredate, wadd, wposition
2) pname, pid, pcost, pzipcode, pstartdate, pduration
3) wid (PK), pid (FK)

3NF:

1) wname, wid, wnum, wsal, whiredate, wadd, wposition
2) pname, pid, pcost, pzipcode, pstartdate, pduration
3) wid (PK), pid (FK)

MANAGER—REPORT—WORKER (*..*)

UNF:

mid, mnum, mhiredate, mname, msal, madd, wname,wid, wnum, wsal, whiredate, wadds, wposition

1NF:

mid, mnum, mhiredate, mname, msal, madd, wname,wid, wnum, wsal, whiredate, wadd, wposition

2NF:

1)mid, mnum, mhiredate, mname, msal, madd
2)wname, wid, wnum, wsal, whiredate, wadd, wposition
3)mid (PK), wid (FK)

3NF:

1)mid, mnum, mhiredate, mname, msal, madd
2)wname, wid, wnum, wsal, whiredate, wadd, wposition
3)mid (PK), wid (FK)

Finalization

1) cid, cname, cadd, chiredate, cnum
2) mid, mnum, mhiredate, mname, msal, madd, cid(FK)
3) nlicense, nfund, nnum, narea
4) mid, mnum, mhiredate, mname, msal, madd, nlicense (FK)
5) ncity, narea
6) pid, pcost, pname, pduration, pstartdate,Penid (FK)
7) Penid, PeNum, Pename
8) wname, wid,wnum,wsal, whiredate, wadd, wposition
9) pname, pid, pcost, pzipCode, pstartdate, pduration
10) wid(PK), pid(FK)
11) mid, mnum, mhiredate, mname, msal, madd
~~12) wname,wid ,wnum,wSal,whiredate,wadd,wposition~~
13) mid(PK), wid(FK)

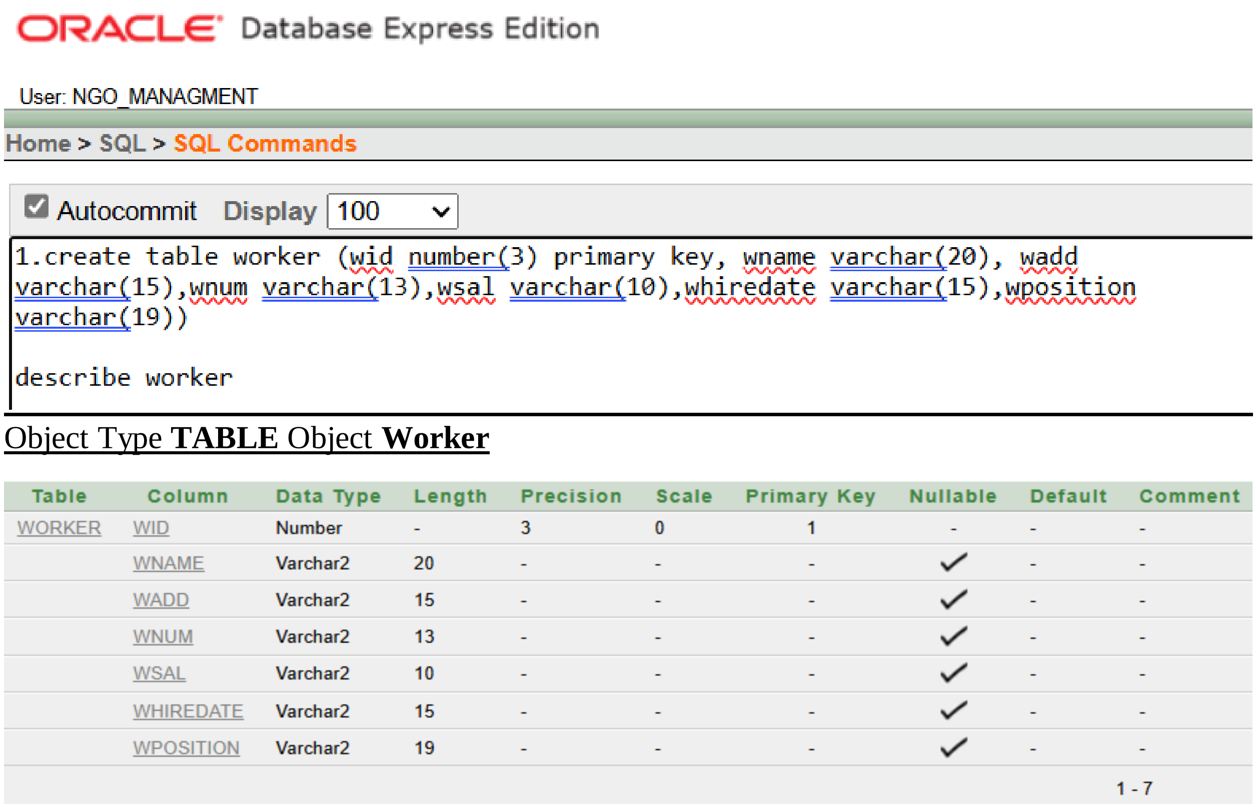
Final Table

1) cid, cname, cadd, chiredate, cnum_ [Customer]
2) mid, mnum, mhiredate, mname, msal, madd, cid(FK) [Inspect]
3) nlicense, nfund, nnum, narea [NGO]
4) mid, mnum, mhiredate, mname, msal, madd, nlicense (FK)[Managed]
5) ncity, narea [NGO_Location]
6) pid, pcost, pname, pduration, pstartdate,Penid (FK) [Serve]
7) Penid, PeNum, Pename [People]
8) wname, wid,wnum,wsal, whiredate, wadd, wposition [Worker]
9) pname, pid, pcost, pzipCode, pstartdate, [Project]
10) wid(PK), pid(FK) [Engage]
11) mid, mnum, mhiredate, mname, msal, madd[Manager]
12) mid(PK), wid(FK)[Report]

Table Creation (DDL Operations)

StudentID1: 23-51584-2 Name: Samaun Islam Ikra	StudentID3: Nazmul Hossain Name: 23-51942-2	
StudentID2: 23-51744-2 Name: Eshtiak Ferdous Mahi	StudentID4: A S M Shahjalal Name: 23-55671-3	
CO4: Creating DML, DDL using Oracle and connection with ODBC/JDBC for existing JAVA application		
PO-e-2: Use modern engineering and IT tools for prediction and modeling of complex computer science and engineering problem		Marks

Worker:



User: NGO_MANAGMENT

Home > SQL > SQL Commands

Autocommit Display 100

```
1.create table worker (wid number(3) primary key, wname varchar(20), wadd
varchar(15),wnum varchar(13),wsal varchar(10),whiredate varchar(15),wposition
varchar(19))

describe worker
```

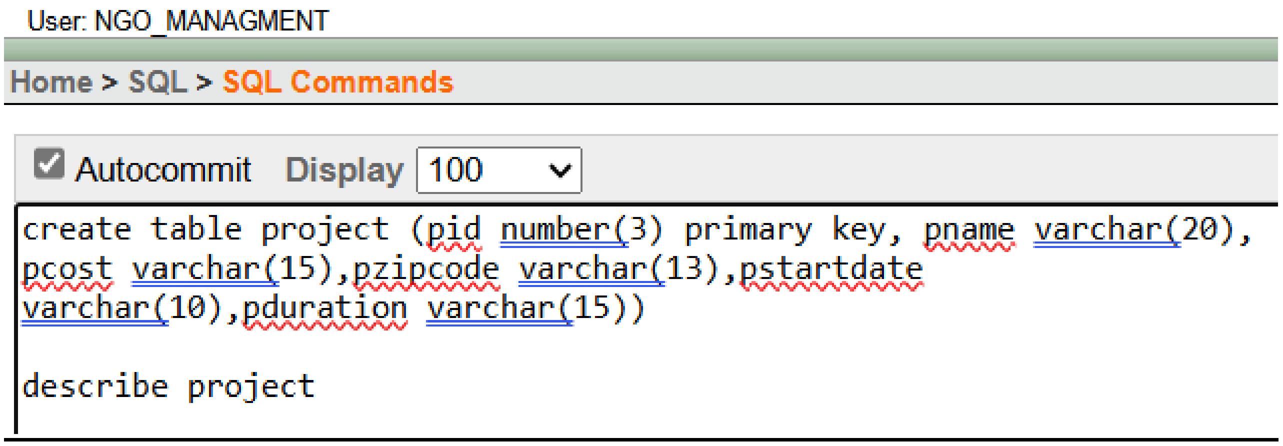
Object Type **TABLE** Object **Worker**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
WORKER	WID	Number	-	3	0	1	-	-	-
	WNAME	Varchar2	20	-	-	-	✓	-	-
	WADD	Varchar2	15	-	-	-	✓	-	-
	WNUM	Varchar2	13	-	-	-	✓	-	-
	WSAL	Varchar2	10	-	-	-	✓	-	-
	WHIREDATE	Varchar2	15	-	-	-	✓	-	-
	WPOSITION	Varchar2	19	-	-	-	✓	-	-

1 - 7

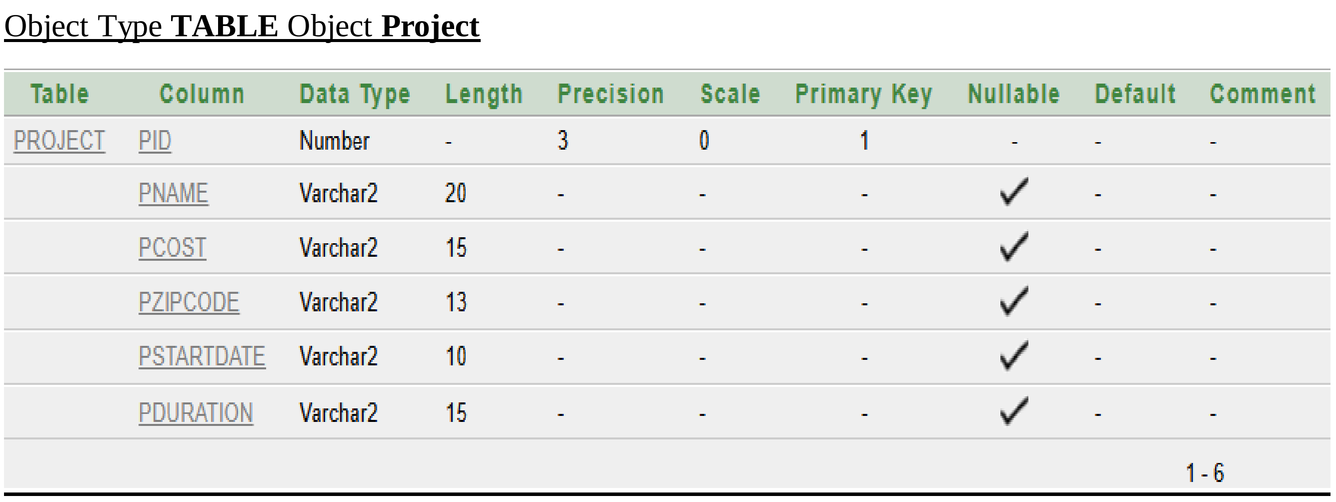
Fig 1: Worker table creation and description

Project:



```
create table project (pid number(3) primary key, pname varchar(20),
pcost varchar(15),pzipcode varchar(13),pstartdate
varchar(10),pduration varchar(15))

describe project
```



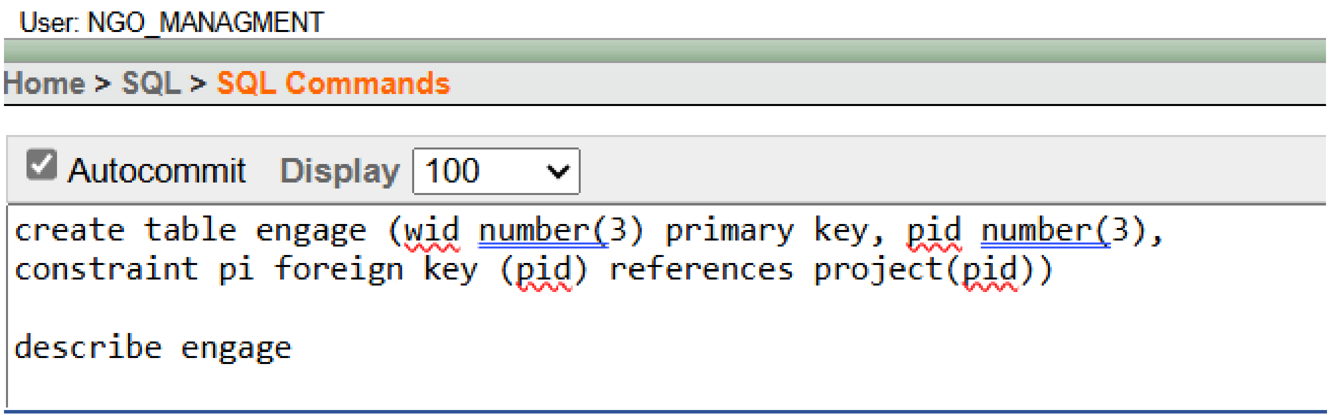
Object Type **TABLE** Object **Project**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PROJECT	PID	Number	-	3	0	1	-	-	-
	PNAME	Varchar2	20	-	-	-	✓	-	-
	PCOST	Varchar2	15	-	-	-	✓	-	-
	PZIPCODE	Varchar2	13	-	-	-	✓	-	-
	PSTARTDATE	Varchar2	10	-	-	-	✓	-	-
	PDURATION	Varchar2	15	-	-	-	✓	-	-

1 - 6

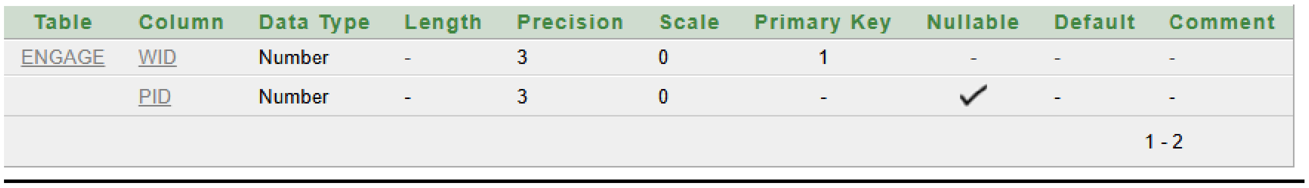
Fig 2: Project table creation and description

Engage:



```
create table engage (wid number(3) primary key, pid number(3),
constraint pi foreign key (pid) references project(pid))

describe engage
```



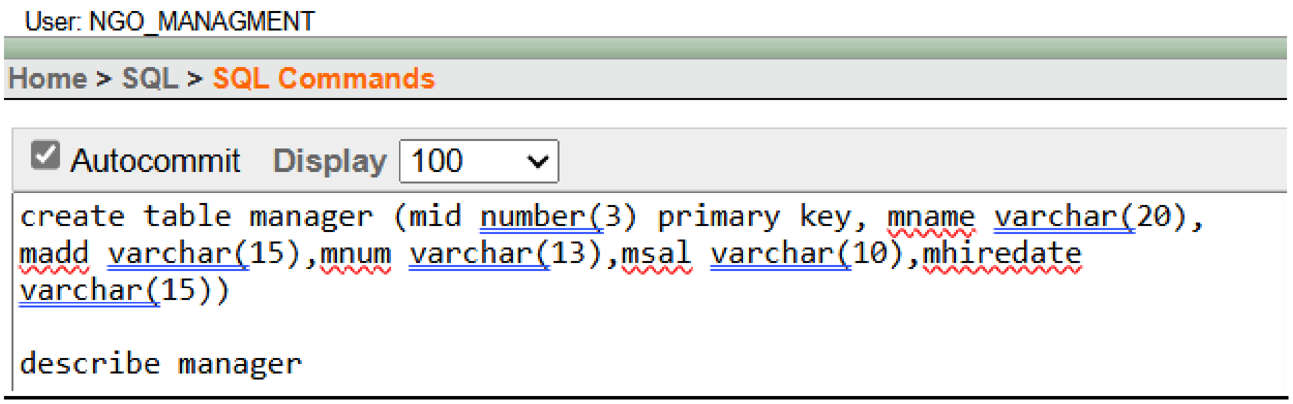
Object Type **TABLE** Object **Engage**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
ENGAGE	WID	Number	-	3	0	1	-	-	-
	PID	Number	-	3	0	-	✓	-	-

1 - 2

Fig 3: Engage table creation and description

Manager:



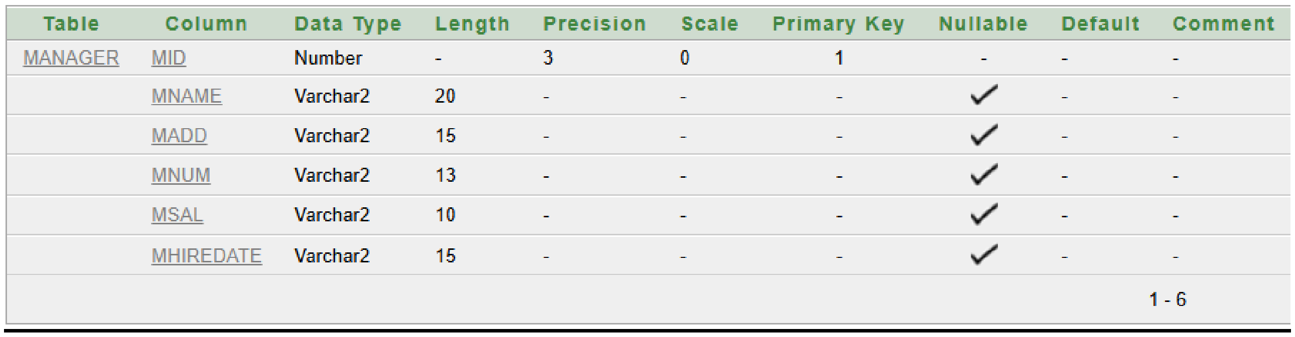
User: NGO_MANAGMENT

Home > SQL > SQL Commands

Autocommit [✓] Display 100

```
create table manager (mid number(3) primary key, mname varchar(20),
madd varchar(15),mnum varchar(13),msal varchar(10),mhiredate
varchar(15))

describe manager
```



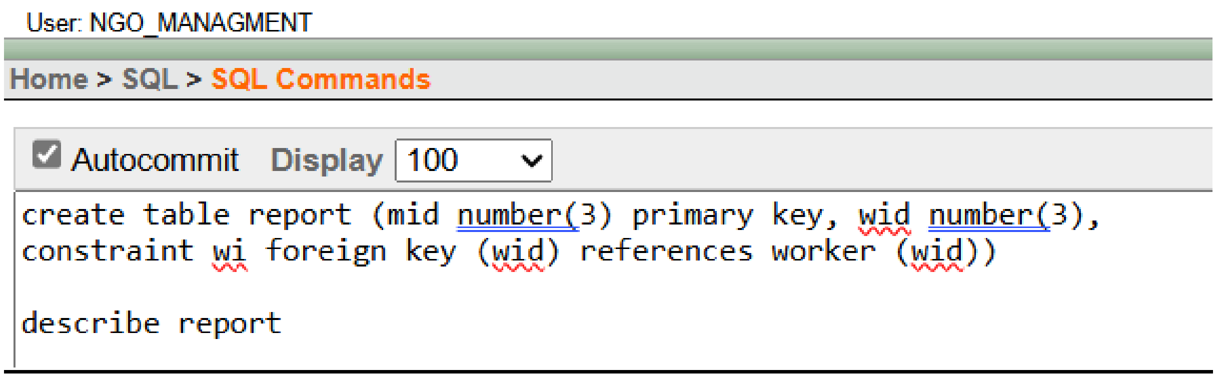
Object Type **TABLE** Object **Manager**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MANAGER	MID	Number	-	3	0	1	-	-	-
	MNAME	Varchar2	20	-	-	-	✓	-	-
	MADD	Varchar2	15	-	-	-	✓	-	-
	MNUM	Varchar2	13	-	-	-	✓	-	-
	MSAL	Varchar2	10	-	-	-	✓	-	-
	MHIREDATE	Varchar2	15	-	-	-	✓	-	-

1 - 6

Fig 4: Manager table creation and description

Report:



```
create table report (mid number(3) primary key, wid number(3),
constraint wi foreign key (wid) references worker (wid))

describe report
```

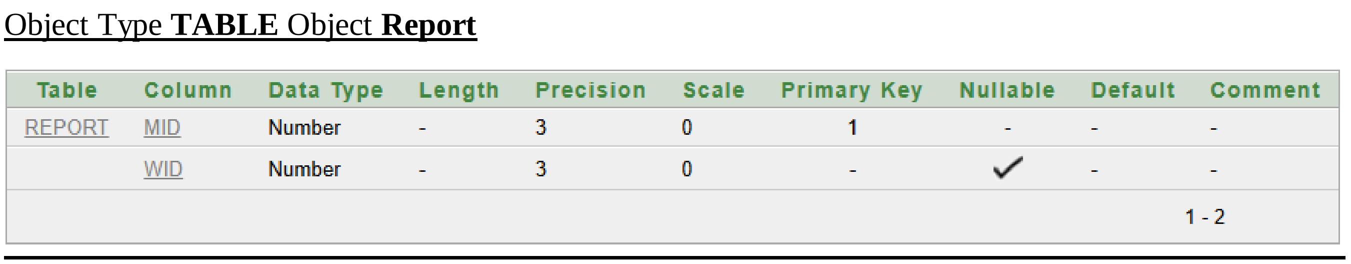
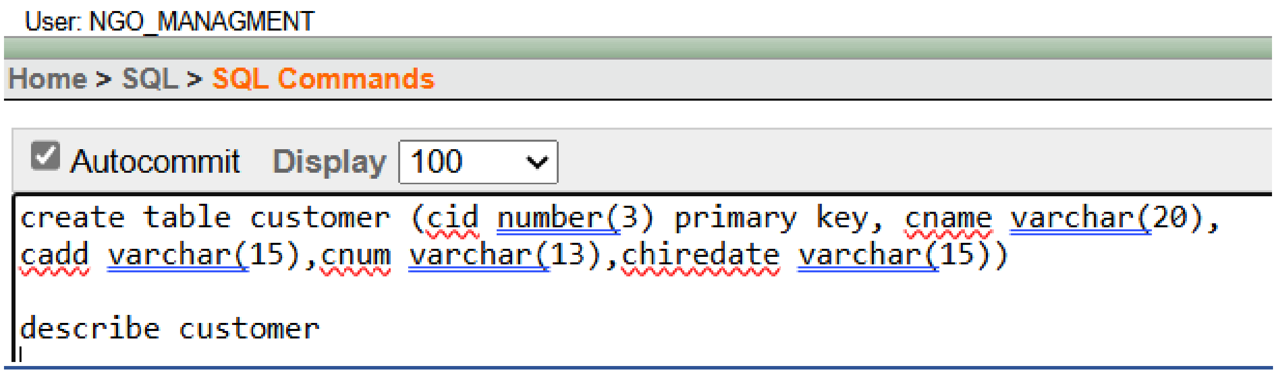


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
REPORT	MID	Number	-	3	0	1	-	-	-
	WID	Number	-	3	0	-	✓	-	-

1 - 2

Fig 5: Report table creation and description

Customer:



```
create table customer (cid number(3) primary key, cname varchar(20),
cadd varchar(15),cnum varchar(13),chiredate varchar(15))

describe customer
```

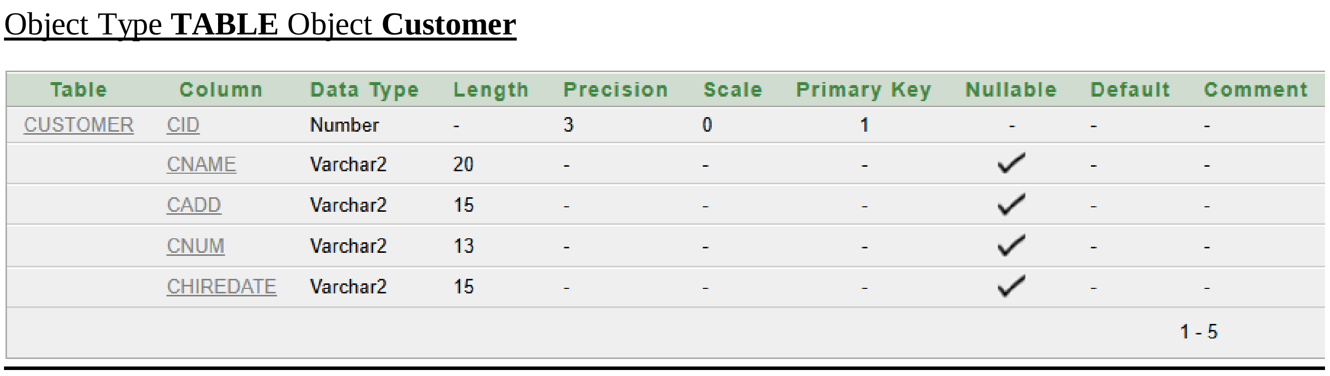
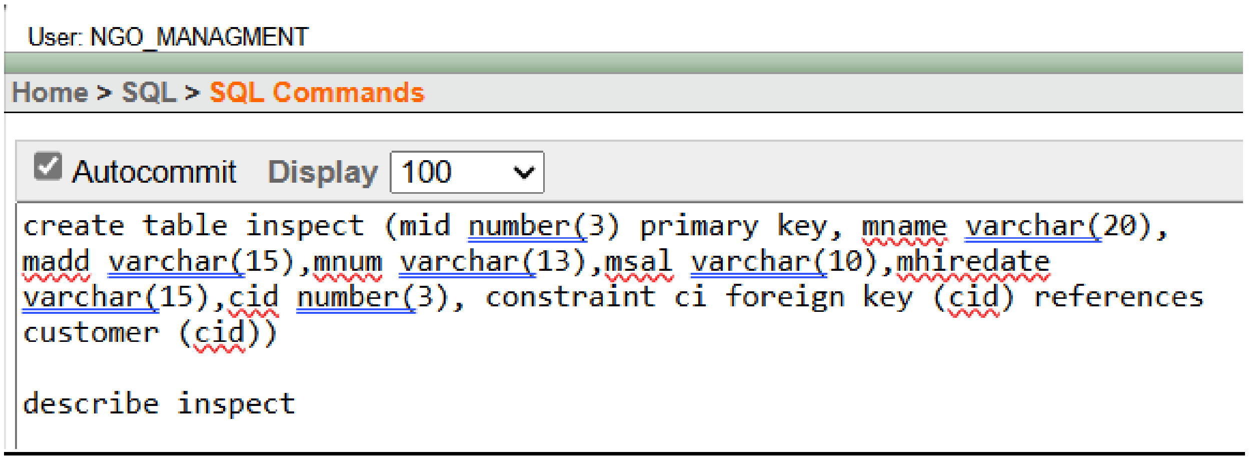


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMER	CID	Number	-	3	0	1	-	-	-
	CNAME	Varchar2	20	-	-	-	✓	-	-
	CADD	Varchar2	15	-	-	-	✓	-	-
	CNUM	Varchar2	13	-	-	-	✓	-	-
	CHIREDATE	Varchar2	15	-	-	-	✓	-	-

1 - 5

Fig 6: Customer table creation and description

Inspect:


User: NGO_MANAGMENT

Home > SQL > SQL Commands

Autocommit Display 100

```
create table inspect (mid number(3) primary key, mname varchar(20),
madd varchar(15),mnum varchar(13),msal varchar(10),mhiredate
varchar(15),cid number(3), constraint ci foreign key (cid) references
customer (cid))

describe inspect
```

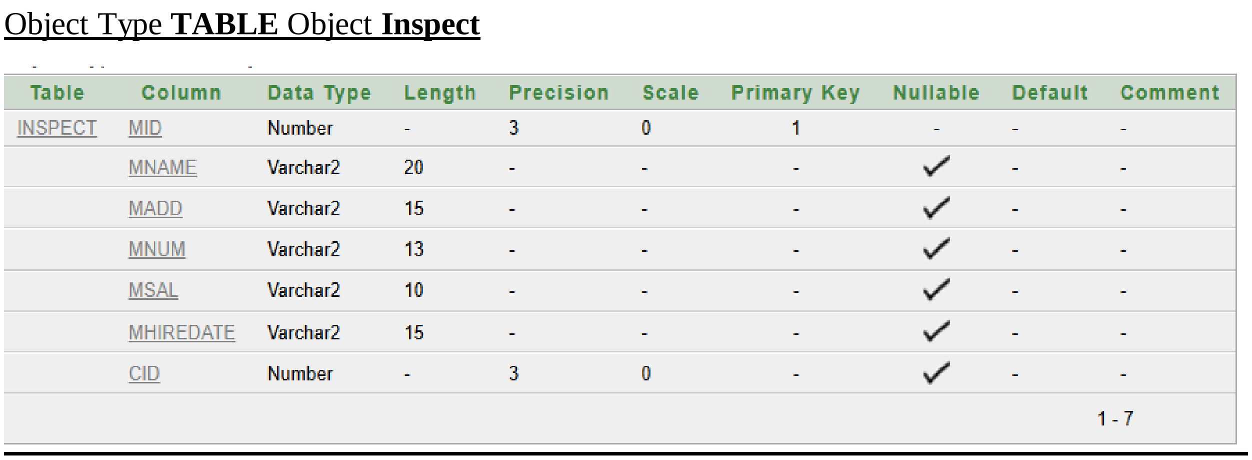
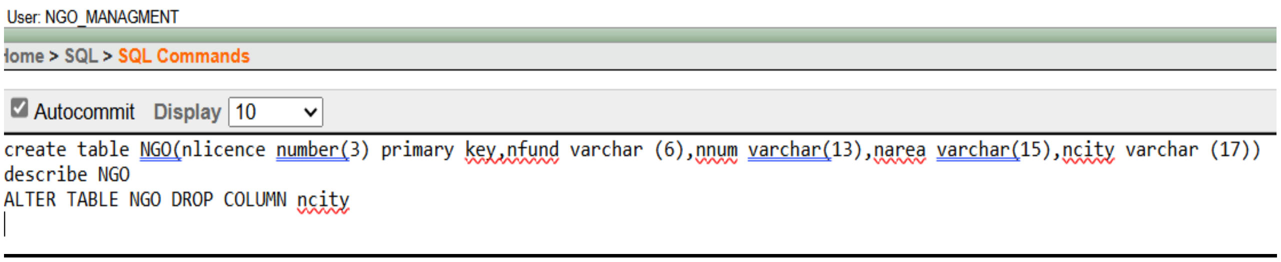

Object Type **TABLE** Object **Inspect**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
INSPECT	MID	Number	-	3	0	1	-	-	-
	MNAME	Varchar2	20	-	-	-	✓	-	-
	MADD	Varchar2	15	-	-	-	✓	-	-
	MNUM	Varchar2	13	-	-	-	✓	-	-
	MSAL	Varchar2	10	-	-	-	✓	-	-
	MHIREDATE	Varchar2	15	-	-	-	✓	-	-
	CID	Number	-	3	0	-	✓	-	-

1 - 7

Fig 7: Inspect table creation and description

NGO:


User: NGO_MANAGMENT

ome > SQL > SQL Commands

Autocommit Display 10

```
create table NGO(nlicence number(3) primary key,nfund varchar (6),nnum varchar(13),narea varchar(15),ncity varchar (17))
describe NGO
ALTER TABLE NGO DROP COLUMN ncity
```

Object Type **TABLE** Object **NGO**

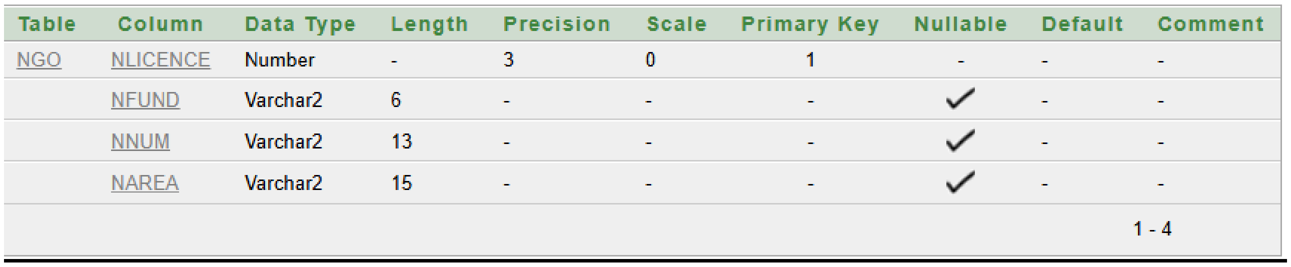
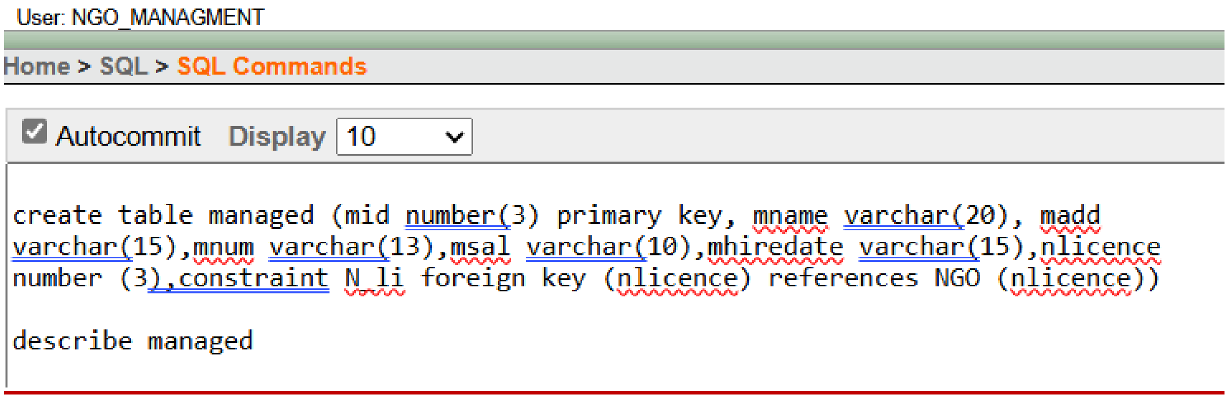


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
NGO	NLICENCE	Number	-	3	0	1	-	-	-
	NFUND	Varchar2	6	-	-	-	✓	-	-
	NNUM	Varchar2	13	-	-	-	✓	-	-
	NAREA	Varchar2	15	-	-	-	✓	-	-

1 - 4

Fig 8: NGO table creation and description

Managed:



User: NGO_MANAGMENT

Home > SQL > SQL Commands

Autocommit Display 10

```
create table managed (mid number(3) primary key, mname varchar(20), madd
varchar(15),mnum varchar(13),msal varchar(10),mhiredate varchar(15),nlicence
number (3),constraint N_li foreign key (nlicence) references NGO (nlicence))

describe managed
```

Object Type **TABLE** Object **Managed**

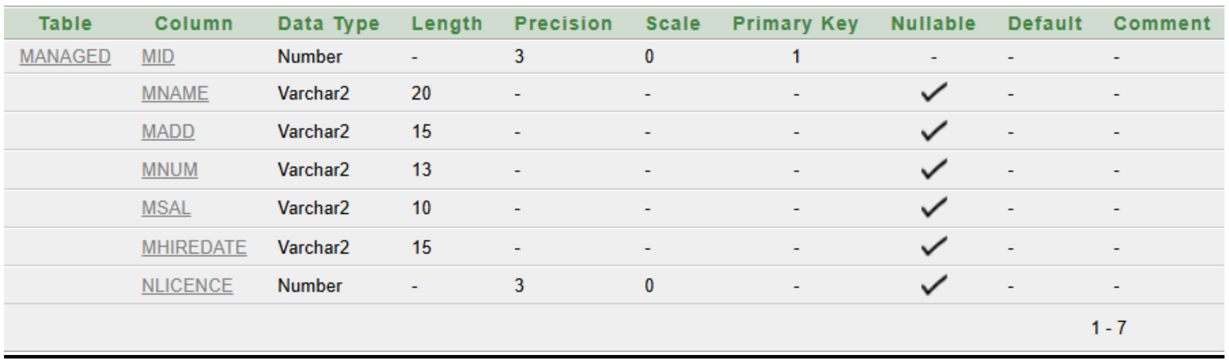
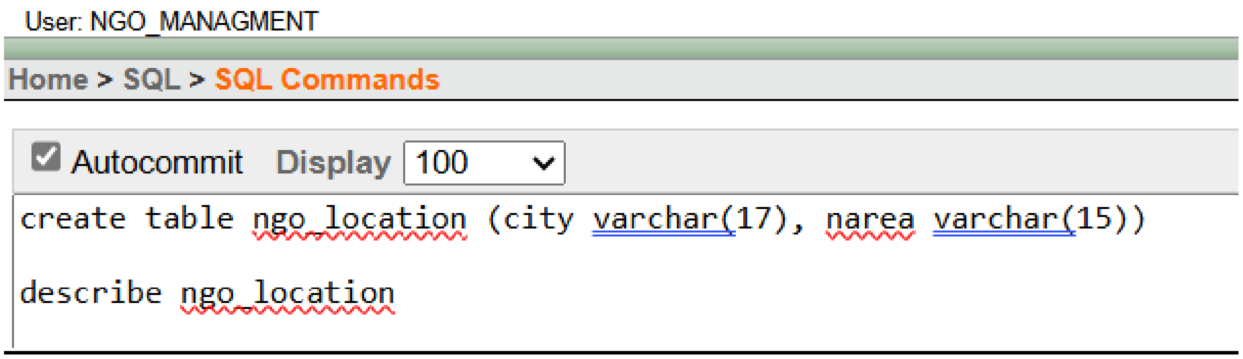


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MANAGED	MID	Number	-	3	0	1	-	-	-
	MNAME	Varchar2	20	-	-	-	✓	-	-
	MADD	Varchar2	15	-	-	-	✓	-	-
	MNUM	Varchar2	13	-	-	-	✓	-	-
	MSAL	Varchar2	10	-	-	-	✓	-	-
	MHIREDATE	Varchar2	15	-	-	-	✓	-	-
	NLICENCE	Number	-	3	0	-	✓	-	-

1 - 7

Fig 9: Managed table creation and description

NGO_location:



```
create table ngo_location (city varchar(17), narea varchar(15))

describe ngo_location
```

Object Type **TABLE** Object **NGO location**

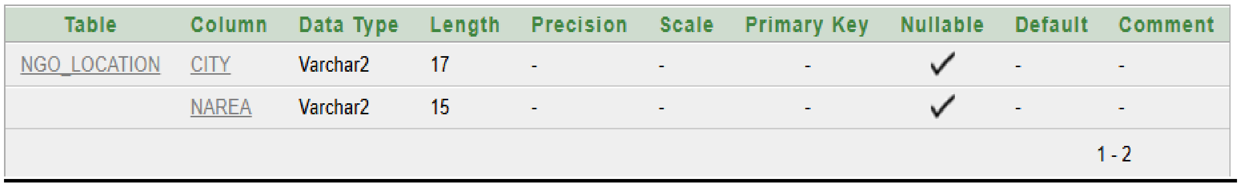
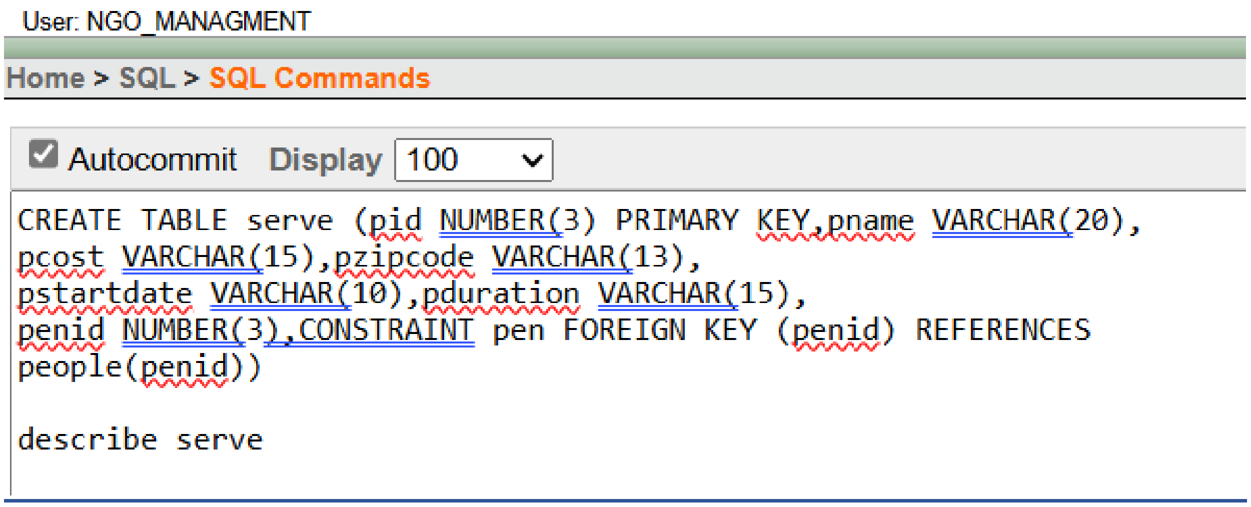


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
NGO_LOCATION	CITY	Varchar2	17	-	-	-	✓	-	-
	NAREA	Varchar2	15	-	-	-	✓	-	-
								1 - 2	

Fig 10: NGO_location table creation and description

Serve:



```
CREATE TABLE serve (pid NUMBER(3) PRIMARY KEY,pname VARCHAR(20),
pcost VARCHAR(15),pzipcode VARCHAR(13),
pstartdate VARCHAR(10),pduration VARCHAR(15),
penid NUMBER(3),CONSTRAINT pen FOREIGN KEY (penid) REFERENCES
people(penid))

describe serve
```

Object Type **TABLE** Object **Serve**

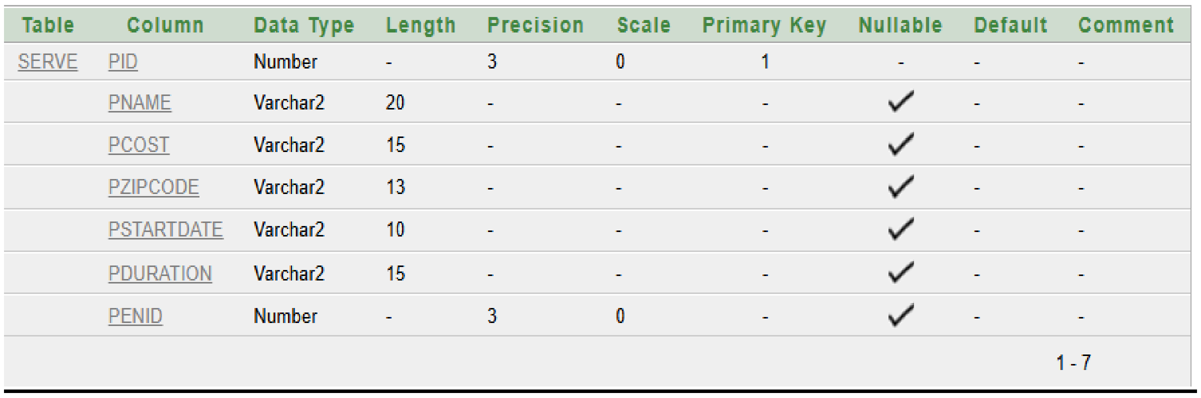
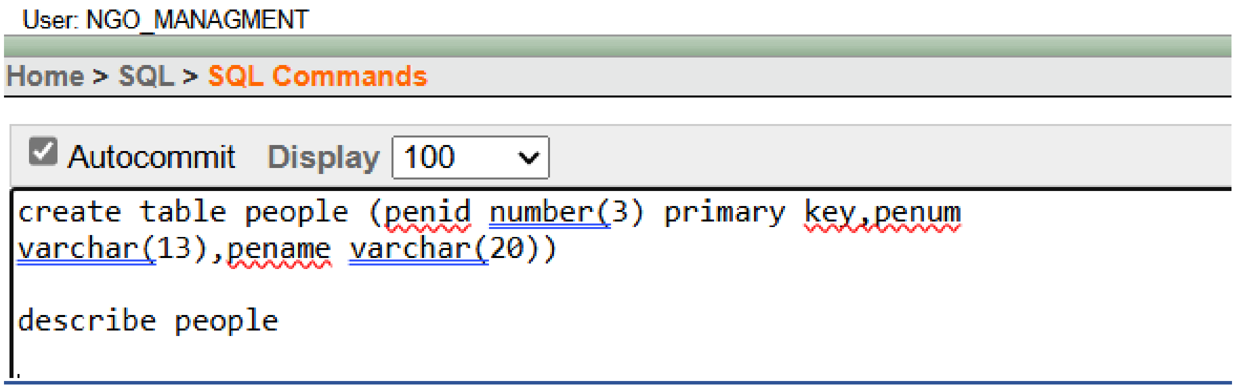


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SERVE	PID	Number	-	3	0	1	-	-	-
	PNAME	Varchar2	20	-	-	-	✓	-	-
	PCOST	Varchar2	15	-	-	-	✓	-	-
	PZIPCODE	Varchar2	13	-	-	-	✓	-	-
	PSTARTDATE	Varchar2	10	-	-	-	✓	-	-
	PDURATION	Varchar2	15	-	-	-	✓	-	-
	PENID	Number	-	3	0	-	✓	-	-

1 - 7

Fig 11: Serve table creation and description

People:



```
create table people (penid number(3) primary key,penum
varchar(13),pename varchar(20))

describe people
```

Object Type **TABLE** Object **People**

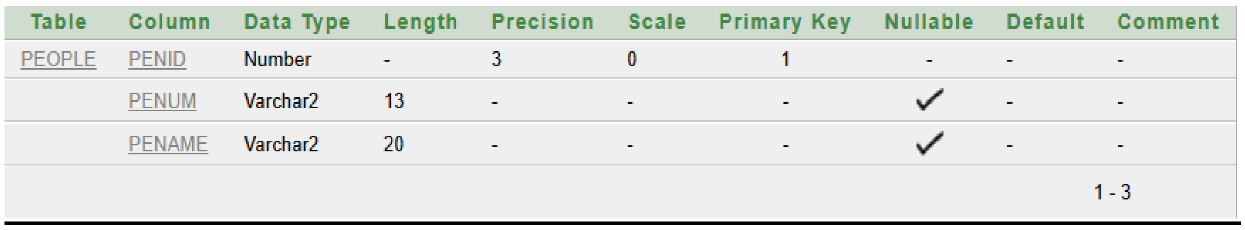
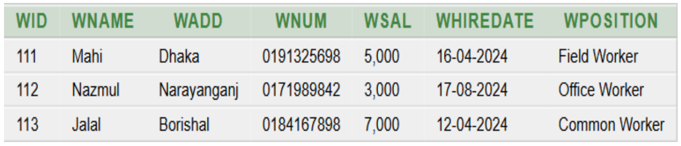


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PEOPLE	PENID	Number	-	3	0	1	-	-	-
	PENUM	Varchar2	13	-	-	-	✓	-	-
	PENAME	Varchar2	20	-	-	-	✓	-	-

1 - 3

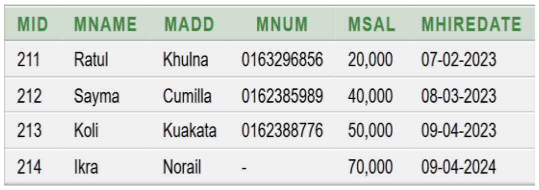
Fig 12: People table creation and description

Inserted Values in the tables



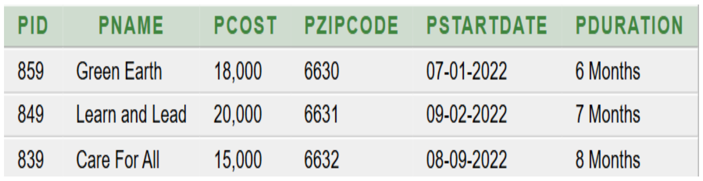
WID	WNAME	WADD	WNUM	WSAL	WHIREDATE	WPOSITION
111	Mahi	Dhaka	0191325698	5,000	16-04-2024	Field Worker
112	Nazmul	Narayanganj	0171989842	3,000	17-08-2024	Office Worker
113	Jalal	Borishal	0184167898	7,000	12-04-2024	Common Worker

Fig 1.1: Worker Table



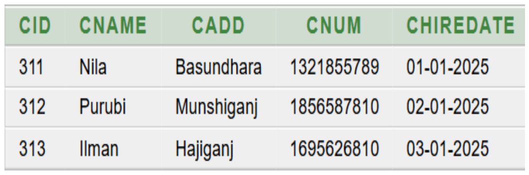
MID	MNAME	MADD	MNUM	MSAL	MHIREDATE
211	Ratul	Khulna	0163296856	20,000	07-02-2023
212	Sayma	Cumilla	0162385989	40,000	08-03-2023
213	Koli	Kuakata	0162388776	50,000	09-04-2023
214	Ikra	Norail	-	70,000	09-04-2024

Fig 4.1: Manager Table



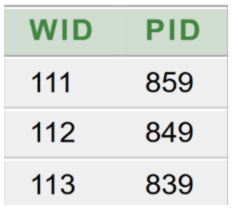
PID	PNAME	PCOST	PZIPCODE	PSTARTDATE	PDURATION
859	Green Earth	18,000	6630	07-01-2022	6 Months
849	Learn and Lead	20,000	6631	09-02-2022	7 Months
839	Care For All	15,000	6632	08-09-2022	8 Months

Fig 2.1: Project Table



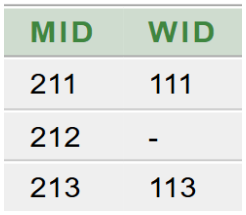
CID	CNAME	CADD	CNUM	CHIREDATE
311	Nila	Basundhara	1321855789	01-01-2025
312	Purubi	Munshiganj	1856587810	02-01-2025
313	Ilman	Hajiganj	1695626810	03-01-2025

Fig 6.1: Customer Table



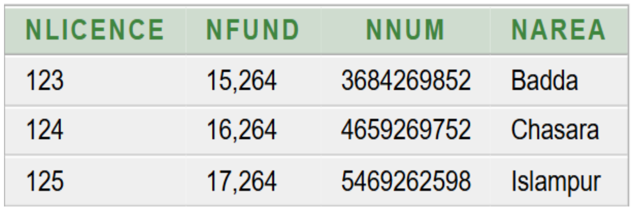
WID	PID
111	859
112	849
113	839

Fig 3.1: Engage Table



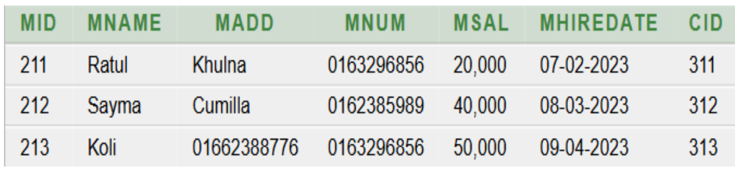
MID	WID
211	111
212	-
213	113

Fig 5.1: Report Table



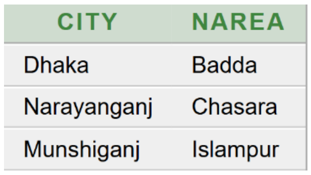
NLICENCE	NFUND	NNUM	NAREA
123	15,264	3684269852	Badda
124	16,264	4659269752	Chasara
125	17,264	5469262598	Islampur

Fig 8.1: NGO Table



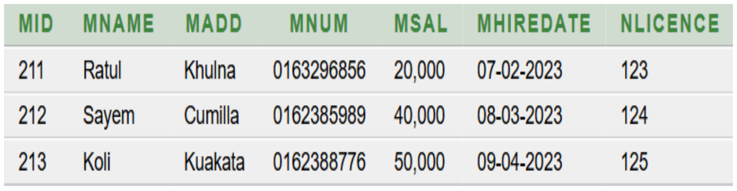
MID	MNAME	MADD	MNUM	MSAL	MHIREDATE	CID
211	Ratul	Khulna	0163296856	20,000	07-02-2023	311
212	Sayma	Cumilla	0162385989	40,000	08-03-2023	312
213	Koli	01662388776	0163296856	50,000	09-04-2023	313

Fig 7.1: Inspect Table



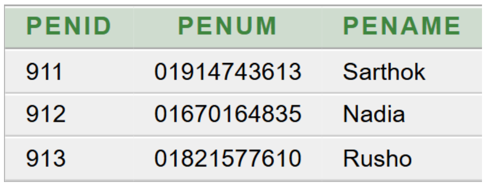
CITY	NAREA
Dhaka	Badda
Narayanganj	Chasara
Munshiganj	Islampur

Fig 10.1: NGO Location Table



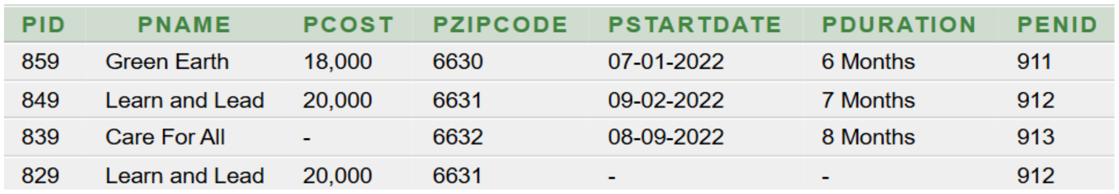
MID	MNAME	MADD	MNUM	MSAL	MHIREDATE	NLICENCE
211	Ratul	Khulna	0163296856	20,000	07-02-2023	123
212	Sayem	Cumilla	0162385989	40,000	08-03-2023	124
213	Koli	Kuakata	0162388776	50,000	09-04-2023	125

Fig 9.1: Managed Table



PENID	PENUM	PENAME
911	01914743613	Sarthok
912	01670164835	Nadia
913	01821577610	Rusho

Fig 12.1: People Table



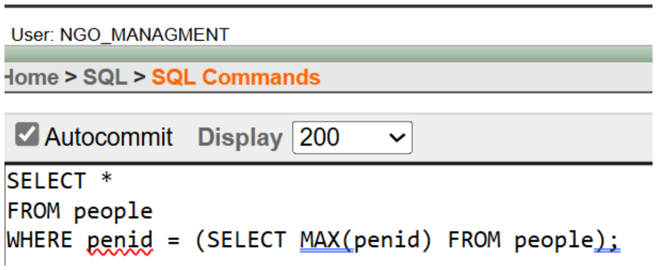
PID	PNAME	PCOST	PZIPCODE	PSTARTDATE	PDURATION	PENID
859	Green Earth	18,000	6630	07-01-2022	6 Months	911
849	Learn and Lead	20,000	6631	09-02-2022	7 Months	912
839	Care For All	-	6632	08-09-2022	8 Months	913
829	Learn and Lead	20,000	6631	-	-	912

Fig 11.1: Serve Table

Query Test in DB

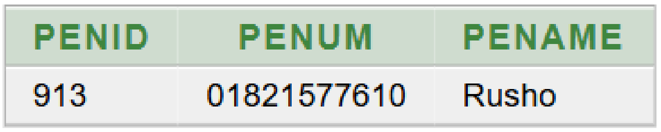
Single Row Subquery

Query: Create a subquery to show all the details of the row in the People table where the penid has the maximum value.



```
SELECT *
FROM people
WHERE penid = (SELECT MAX(penid) FROM people);
```

Fig: Single row subquery creation command

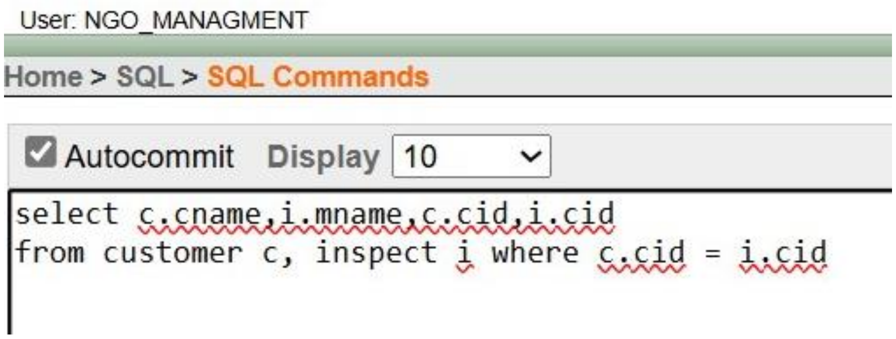


PENID	PENUM	PENAME
913	01821577610	Rusho

Fig: Output of single row subquery as a whole table

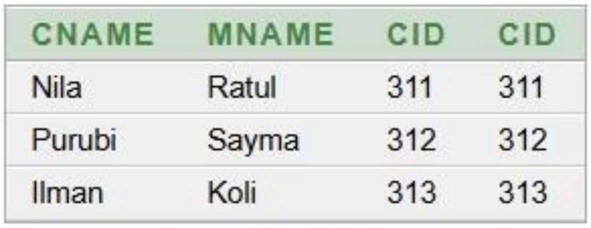
Equijoin

Query: For each Customer, list the Customer name,ID and Manager, Manager name to retrieve and also match data from the two tables.



```
select c.cname,i.mname,c.cid,i.cid
from customer c, inspect i where c.cid = i.cid
```

Fig: Equijoin creation command

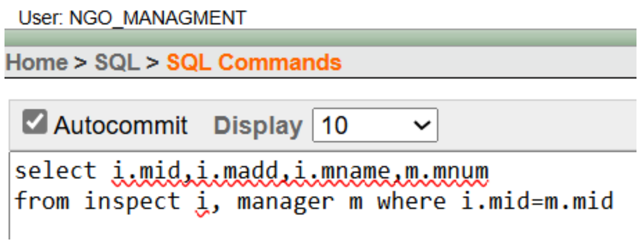


CNAME	MNAME	CID	CID
Nila	Ratul	311	311
Purubi	Sayma	312	312
Ilman	Koli	313	313

Fig: Output of Equijoin as a whole table

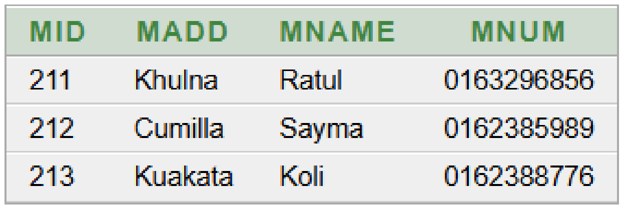
NON-Equijoin

Query: Find the mid, madd, and mname from the inspect table and the mnum from the manager table where the mid in the inspect table is not equal to the mid in the manager table.



```
select i.mid,i.madd,i.mname,m.mnum
from inspect i, manager m where i.mid=m.mid
```

Fig: Non-Equijoin creation command

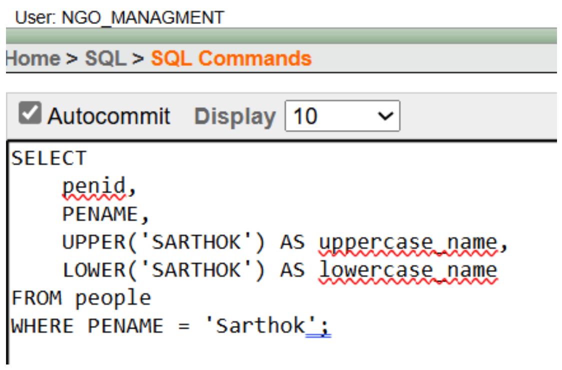


MID	MADD	MNAME	MNUM
211	Khulna	Ratul	0163296856
212	Cumilla	Sayma	0162385989
213	Kuakata	Koli	0162388776

Fig: Output of Non-Equijoin as a whole table

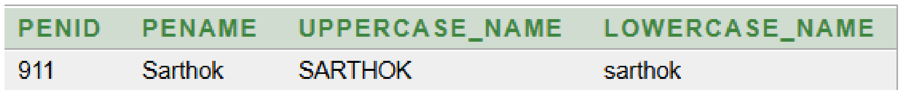
Single Row Function

Query: Show to retrieve the penid and PENAME for the row where the name is Sarthok, along with the uppercase and lowercase representations of the string 'SARTHOK'**.**



```
SELECT
    penid,
    PENAME,
    UPPER('SARTHOK') AS uppercase_name,
    LOWER('SARTHOK') AS lowercase_name
FROM people
WHERE PENAME = 'Sarthok';
```

Fig: Single Row Function creation command

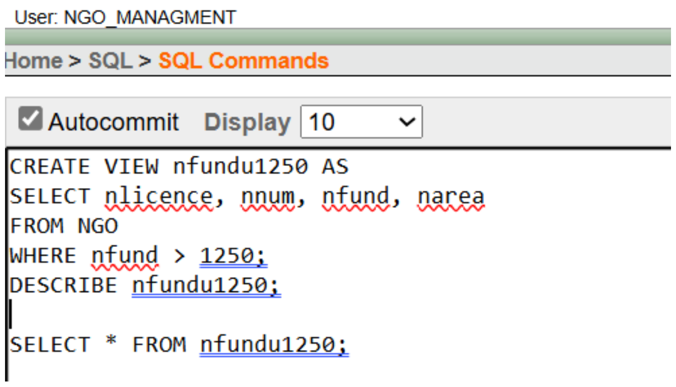


PENID	PENAME	UPPERCASE_NAME	LOWERCASE_NAME
911	Sarthok	SARTHOK	sarthok

Fig: Output of Single Row Function as a whole table

Simple view

Query: Show to display the details (license, number, fund, and area) of NGOs where the fund exceeds 1250



User: NGO_MANAGMENT

Home > SQL > SQL Commands

Autocommit ✓ Display 10

```
CREATE VIEW nfundu1250 AS
SELECT nlicence, nnum, nfund, narea
FROM NGO
WHERE nfund > 1250;
DESCRIBE nfundu1250;

SELECT * FROM nfundu1250;
```

Fig: Simple view creation command

Object Type VIEW Object **nfund1250**

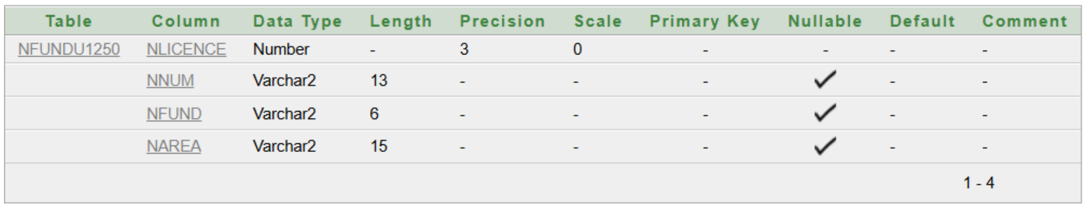


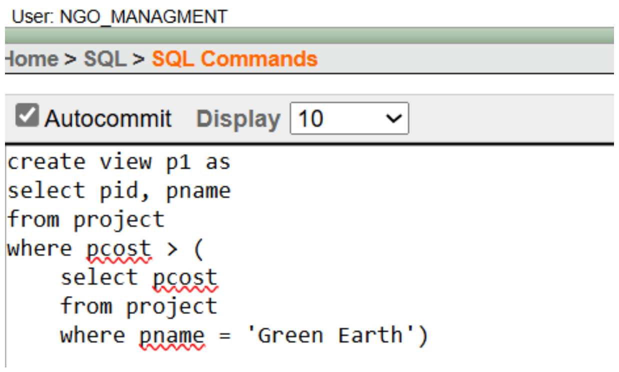
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
NFUNDU1250	NLICENCE	Number	-	3	0	-	-	-	-
	NNUM	Varchar2	13	-	-	-	✓	-	-
	NFUND	Varchar2	6	-	-	-	✓	-	-
	NAREA	Varchar2	15	-	-	-	✓	-	-

1 - 4

Fig: Description of the Simple view

Complex View

Query: Display complex view named as project where the PID, PNAME will be shown where the pcost is greater when PNAME is Green Earth.



User: NGO_MANAGMENT

Home > SQL > SQL Commands

Autocommit ✓ Display 10

```
create view p1 as
select pid, pname
from project
where pcost > (
    select pcost
    from project
    where pname = 'Green Earth')
```

Fig: Complex View creation command

Object Type VIEW Object **P1**

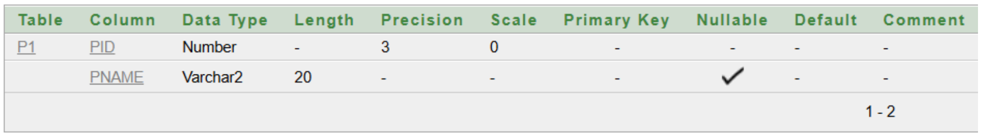
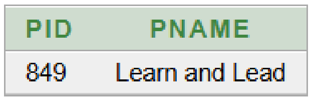


Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
P1	PID	Number	-	3	0	-	-	-	-
	PNAME	Varchar2	20	-	-	-	✓	-	-

1 - 2

Fig: Description of the Complex view

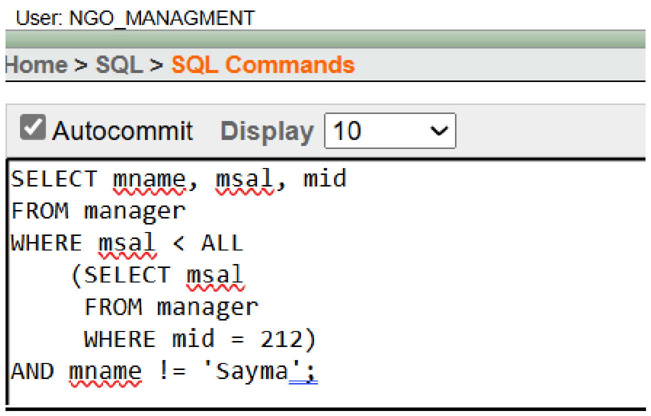


PID	PNAME
849	Learn and Lead

Fig: Output of the Complex View as a whole table

Multiple Row Subquery

Query: Write a Multiple row Subquery to display the mname, msal , and mid of all manager whose msal are less than any product with the mid 212, excluding the product named 'Sayma'.



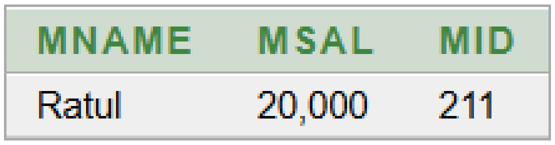
User: NGO_MANAGMENT

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT mname, msal, mid
FROM manager
WHERE msal < ALL
    (SELECT msal
     FROM manager
     WHERE mid = 212)
AND mname != 'Sayma';
```

Fig: Multiple Row Subquery creation command

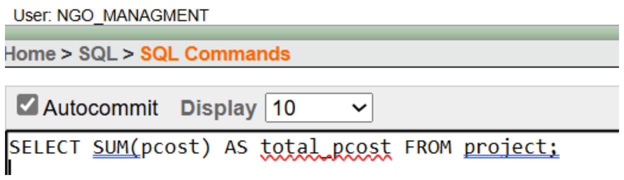


MNAME	MSAL	MID
Ratul	20,000	211

Fig: Output of the Multiple Row Subquery as a whole table

Multiple Function

Query: To calculate the total_cost of all pcost in the project table.



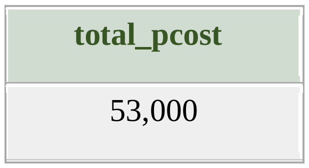
User: NGO_MANAGMENT

Home > SQL > SQL Commands

Autocommit Display 10

```
SELECT SUM(pcost) AS total_pcost FROM project;
```

Fig: Multiple Function creation command



total_pcost
53,000

Fig: Output of the Multiple Function as a whole table

Description of a Successful DB connection

StudentID1: 23-51584-2	Name: Samaun Islam Ikra

The Code shows a Java program in Apache NetBeans IDE 24 that connects to an Oracle database using JDBC. Below is an explanation of the tools used, and steps taken:

Tools Used:

1. Apache NetBeans IDE 24
    - An integrated development environment (IDE) for Java development.
    - Provides features like syntax highlighting, debugging, and code execution.
2. Oracle Database
    - A lightweight, free-to-use Oracle database used for development and testing.
    - The connection string (jdbc:oracle:thin:@localhost:1521:xe) indicates the database running locally on port 1521.
3. JDBC (Java Database Connectivity)
    - A Java API for connecting to databases and executing SQL queries.
    - The driver oracle.jdbc.driver.OracleDriver is used for communication between Java and Oracle.

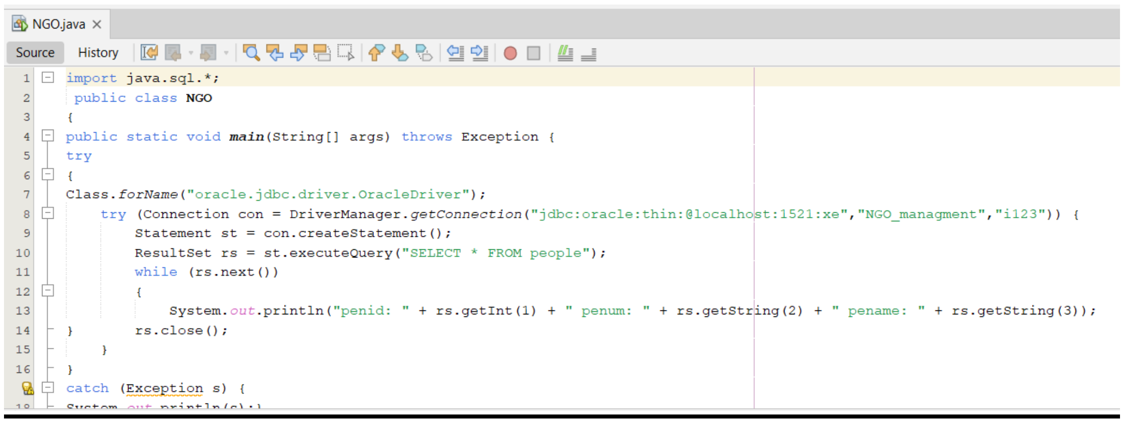
Steps Explained:

1. Import SQL Package
   - This imports JDBC classes for database connection and query execution.
2. Load Oracle JDBC Driver
   - Ensures the Oracle JDBC driver is available for use.
3. Establish Connection to Oracle Database
   - Connects to the NGO_managment database using username NGO_managment and password i123.
4. Create and Execute SQL Statement
   - Creates a Statement object to execute the SQL query.
   - Runs "SELECT * FROM people" to fetch all records from the people table.
5. Process Query Results
   - Iterates through the ResultSet, extracting and printing the penid, penum, and pename columns.
6. Close Database Connection
   - Closes the ResultSet to free database resources.
7. Handle Exceptions
   - Catches and prints any database connection or SQL execution errors.

Next Steps:

- Ensure the Oracle JDBC driver (ojdbc.jar) is added to the NetBeans project.
- Verify that the Oracle database service is running on localhost:1521.
- Confirm that the people table exists in the **NGO_managment** schema.
- Run the program and check for successful database connection and data retrieval.

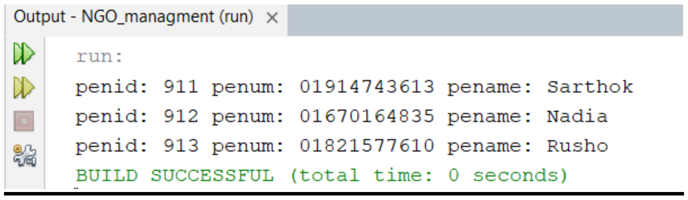
Java Code:



```java
import java.sql.*;
 public class NGO
{
public static void main(String[] args) throws Exception {
try
{
Class.forName("oracle.jdbc.driver.OracleDriver");
    try (Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","NGO_managment","i123")) {
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT * FROM people");
        while (rs.next())
        {
            System.out.println("penid: " + rs.getInt(1) + " penum: " + rs.getString(2) + " pename: " + rs.getString(3));
}       rs.close();
    }
}
catch (Exception s) {
System.out.println(s);
```

Fig: JAVA code for Database connection

The Output:



```
run:
penid: 911 penum: 01914743613 pename: Sarthok
penid: 912 penum: 01670164835 pename: Nadia
penid: 913 penum: 01821577610 pename: Rusho
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig: Output After running the code

Name-Eshtiak Fardous Mahi	ID-23-51744-2

The Java code provided is a JDBC (Java Database Connectivity) program that connects to an Oracle Database, executes a SQL SELECT query, retrieves data from the project table, and prints it to the console.

Tools Used:

1. **Java** – The programming language used to write the program.
2. **JDBC (Java Database Connectivity)** – A Java API that enables interaction with databases.
3. **Oracle Database** – A relational database management system (RDBMS) where the "project" table is stored.
4. **Oracle JDBC Driver** – A Java library (oracle.jdbc.driver.OracleDriver) that allows Java applications to communicate with an Oracle database.

<u>Step by this code:</u>

1. Load the Oracle JDBC Driver

- This **dynamically loads** the Oracle JDBC driver into memory.
- It enables Java to interact with Oracle Database.

2. Establish a Connection to the Database

- establishes a connection to the database.
- Connection String Explanation
- "NGO_MANAGMENT" – The **database username**.
- "i123" – The **database password**.

3. Create a SQL Statement

- Creates a **Statement** object (st) to execute SQL queries.

4. Execute SQL Query and Retrieve Data

- Executes a **SQL query** (SELECT * FROM project) to fetch all records from the **project** table.
- The results are stored in the **ResultSet** object (rs).

5. Process the Query Result

- Iterates through each row in the result set.
- rs.getInt("pid") – Retrieves the **Project ID** (Integer).
- rs.getString("pname") – Retrieves the **Project Name** (String).
- rs.getString("pcost") – Retrieves the **Project Cost** (String).
- rs.getString("pzipcode") – Retrieves the **Zipcode** (String).
- rs.getString("pstartdate") – Retrieves the **Project Start Date** (String).
- rs.getString("pduration") – Retrieves the **Project Duration** (String).
- Print the **retrieved data** to the console.

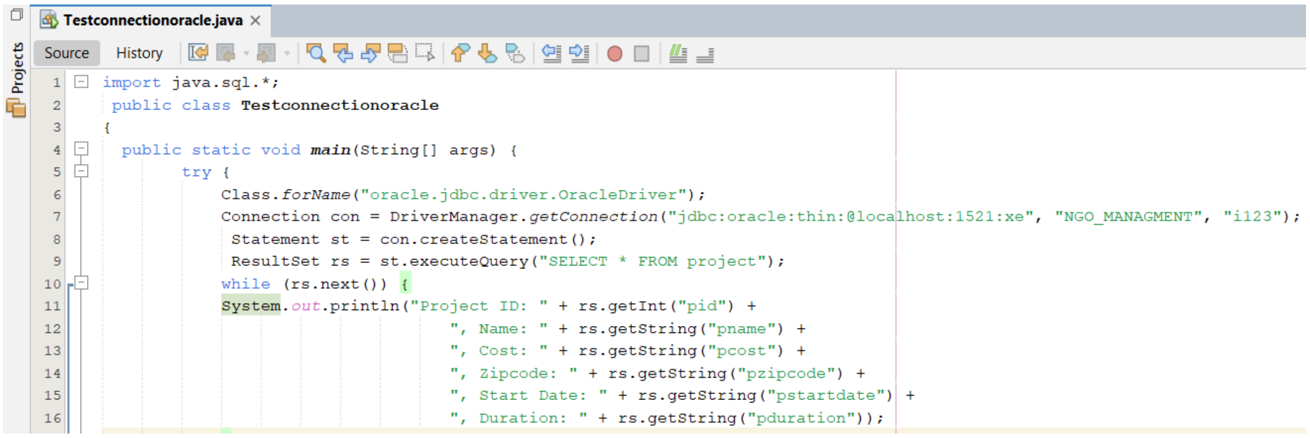
6. Close Resources

- **rs.close();** – Closes the ResultSet object (frees up memory).
- **st.close();** – Closes the Statement object.
- **con.close();** – Closes the **database connection** to prevent memory leaks.

7. Handle Exceptions

- **Catches any errors** that occur while executing the database operations.
- Prints the **error message** to the console if something goes wrong.

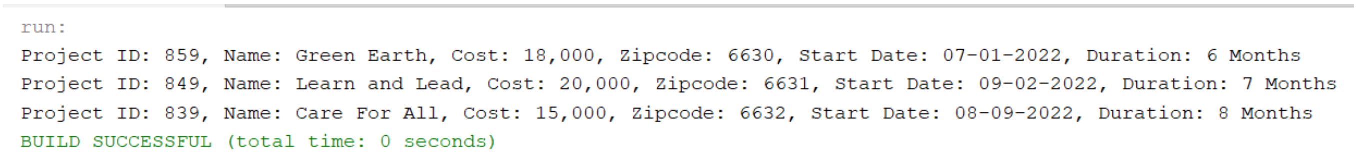
Java Code:



```java
import java.sql.*;
 public class Testconnectionoracle
{
  public static void main(String[] args) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "NGO_MANAGMENT", "i123");
             Statement st = con.createStatement();
             ResultSet rs = st.executeQuery("SELECT * FROM project");
            while (rs.next()) {
            System.out.println("Project ID: " + rs.getInt("pid") +
                                   ", Name: " + rs.getString("pname") +
                                   ", Cost: " + rs.getString("pcost") +
                                   ", Zipcode: " + rs.getString("pzipcode") +
                                   ", Start Date: " + rs.getString("pstartdate") +
                                   ", Duration: " + rs.getString("pduration"));
```

Fig: JAVA code for Database connection

The Output:



```
run:
Project ID: 859, Name: Green Earth, Cost: 18,000, Zipcode: 6630, Start Date: 07-01-2022, Duration: 6 Months
Project ID: 849, Name: Learn and Lead, Cost: 20,000, Zipcode: 6631, Start Date: 09-02-2022, Duration: 7 Months
Project ID: 839, Name: Care For All, Cost: 15,000, Zipcode: 6632, Start Date: 08-09-2022, Duration: 8 Months
BUILD SUCCESSFUL (total time: 0 seconds)
```

Fig: Output After running the code

Conclusion

The **NGO Management System** is a centralized platform designed to streamline and optimize the operations of non-governmental organizations. It offers tools for managing volunteer registration, event planning, donor tracking, fund allocation, beneficiary records, and detailed reporting. The system ensures transparency, efficiency, and effective communication between stakeholders, helping NGOs make a meaningful impact. With features like a user-friendly interface, automated workflows, real-time data insights, and secure storage, it is tailored to meet the unique needs of NGOs while enhancing resource allocation and operational effectiveness.

Looking to the future, the system aims to integrate emerging technologies such as AI for predicting donation trends and beneficiary needs, blockchain for secure and transparent fund management, and IoT devices for real-time resource tracking. Mobile application development is also planned to ensure greater accessibility for volunteers, donors, and beneficiaries, including offline functionality for remote areas. The system will expand globally, offering multilingual and multi-currency support while adapting to local regulations and cultural contexts. Advanced analytics, predictive insights, and interactive dashboards will further enhance decision-making. Future features include sustainability tools to track environmental impact, crowdfunding integration, enhanced collaboration tools, and gamified experiences to boost engagement. Open-source collaboration will invite global contributions, fostering adaptability and innovation in NGO operations.