AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

Course Title

INTRODUCTION TO DATABASE

Fall 2023-2024
Section: G

TITLE

TRIPIFY – A TRAVEL & TOURISAM MANAGEMENT SYSTEM

Supervised By

MD Sajid Bin Faisal

Submitted By: Group no: 03

Name	ID
SHAILY SAHA	22-48530-3
PRETOM CHANDRA ROY	22-48556-3
TURJO DAS DIP	22-48558-3

TABLE OF CONTENTS

TOPICS		Page no.
Title Page		1
Table of Content		2
1.	Introduction	3
2.	Case Study	3
3.	ER Diagram	4
4.	Normalization	5-6
5.	Finalization	7
6.	Table Creation	8-13
7.	Data Insertion	14-16
8.	Query Test	17-25
9.	Database Connection	26-29
10.	Conclusion	30

1.Introduction:

The title of the project is "**Tripify – A Travel and Tourism Management System**". By using the features of Oracle 10g, this system offers seamless booking, transport management, transport, payment processing to make travel planning convenient for users. With a user-friendly interface, it aims to provide a hassle-free experience for travelers. We had to go through case study, ER diagram, the normalization process for each relation, finalization and final tables and value insertion. By integrating key features, the project embraces the future of travel management with our innovative solution.

2.Case Study:

In Tripify - A Travel and Tourism management system, A customer is identified by customer id. The system contains name, address, email, mobile number, and national id. A customer address is composed of house number, street name and city. A customer name is also composed of first name and last name. Firstly, A customer reserves a hotel. One hotel must be reserved by one customer and one customer cannot reserve more than one hotel. The hotel is identified by hotel Id. It also contains contact number, hotel name, hotel license number, hotel location and hotel cost. The amount of reservation is stored into the system. Customer books package. A customer must book a package and a customer cannot book more than one package. Package is identified by Package Id. It also contains Cost, Package name and Destinations. The amount of booking a package is stored into the system. Customer also schedules transport. Customer must schedule a transport and Customer cannot schedule more than once. Transport is identified by transport no. The system also stores ticket cost, arrival time, departure time, departure location and arrival location. Transport is generalized by train, bus, and flight. The amount of scheduling a transport is also stored into the system. After finishing the booking process, the customer makes payment. Payment is generalized by card payment and Mobile banking. Mobile banking is generalized by Nogod and Bkash. Card payments stores card number, card name, expiration date and cvc no. Payment system stores Payment id and amount. Lastly, Customer gives review. which is identified by review id, rating, comments, date.

3.Er Diagram:

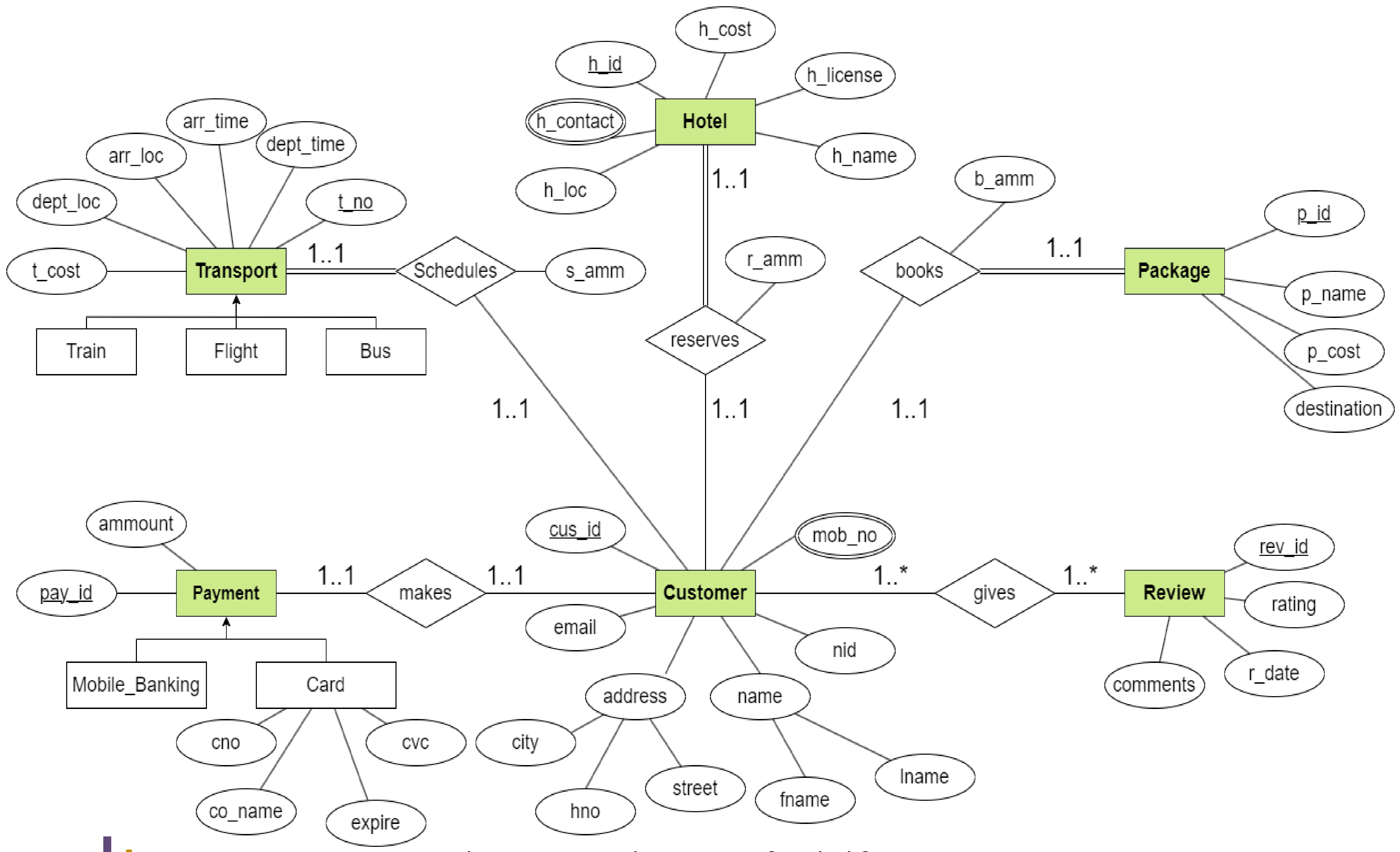


Figure: Er Diagram of Tripify

4.Normalization:

Reserve Relation:

UNF:
cus_id,name,fname,lname,nid,address,hno,city,street,mob_no,email,h_id,h_name,h_loc,

h_contact, h_license,h_cost

1NF:
cus_id,name,fname,lname,nid,address,hno,city,street,mob_no,email,h_id,h_name,h_loc,

h_contact, h_license,h_cost

2NF:

1. cus_id (PK), fname, lname, nid, hno, city, street, mob_no, email
2. h_id (PK), h_name, h_loc, h_contact, h_license,h_cost
3. cus_id (PK), h_id (FK),r_amm

3NF: Same as 2NF

Schedules Relation:

UNF: cus_id, name, fname, lname, nid, address, hno, city, street, mob_no, email, t_no, dept_loc, arr_loc, dept_time, arr_time, t_cost

1NF: cus_id, name, fname, lname, nid, address, hno, city, street, mob_no, email, t_no, dept_loc, arr_loc, dept_time, arr_time, t_cost

2NF:

1. t_no(PK), dept_loc, arr_loc, dept_time, arr_time, t_cost
2. cus_id (PK), fname, lname, nid, hno, city, street, mob_no, email
3. cus_id (PK), t_no(FK),s_amm

3NF: Same as 2NF

Books Relation:

UNF: cus_id, name, fname, lname, nid, address, hno, city, street, mob_no, email, p_id, p_name, destination, p_cost

1NF: cus_id, name, fname, lname, nid, address, hno, city, street, mob_no, email, p_id, p_name, destination, p_cost

2NF:

1. p_id(PK), p_name, destination, p_cost
2. cus_id (PK), fname, lname, nid, hno, city, street, mob_no, email
3. cus_id (PK), p_id(FK),b_amm

3NF: Same as 2NF

Makes Relation:

UNF: cus_id, name, fname, lname, nid, address, hno, city, street, mob_no, email, pay_id, amount, co_name, c_no, cvc, expire.

1NF: cus_id, name, fname, lname, nid, address, hno, city, street, mob_no, email, pay_id, amount, co_name, c_no, cvc, expire.

2NF:

1. cus_id (PK), fname, lname, nid, hno, city, street, mob_no, email
2. pay_id(PK), amount, co_name, c_no, cvc, expire
3. cus_id (PK), pay_id (FK)

3NF: Same as 2NF

Gives Relation:

UNF: cus_id, name, fname, lname, nid, address, hno, city, street, mob_no, email, rev_id, rating, r_date, comment

1NF: cus_id, name, fname, lname, nid, address, hno, city, street, mob_no, email, rev_id, rating, r_date, comment

2NF:

1. cus_id (PK), fname, lname, nid, hno, city, street, mob_no, email
2. rev_id(PK), rating, r_date, comment
3. cus_id (PK), rev_id (FK)

3NF: Same as 2NF

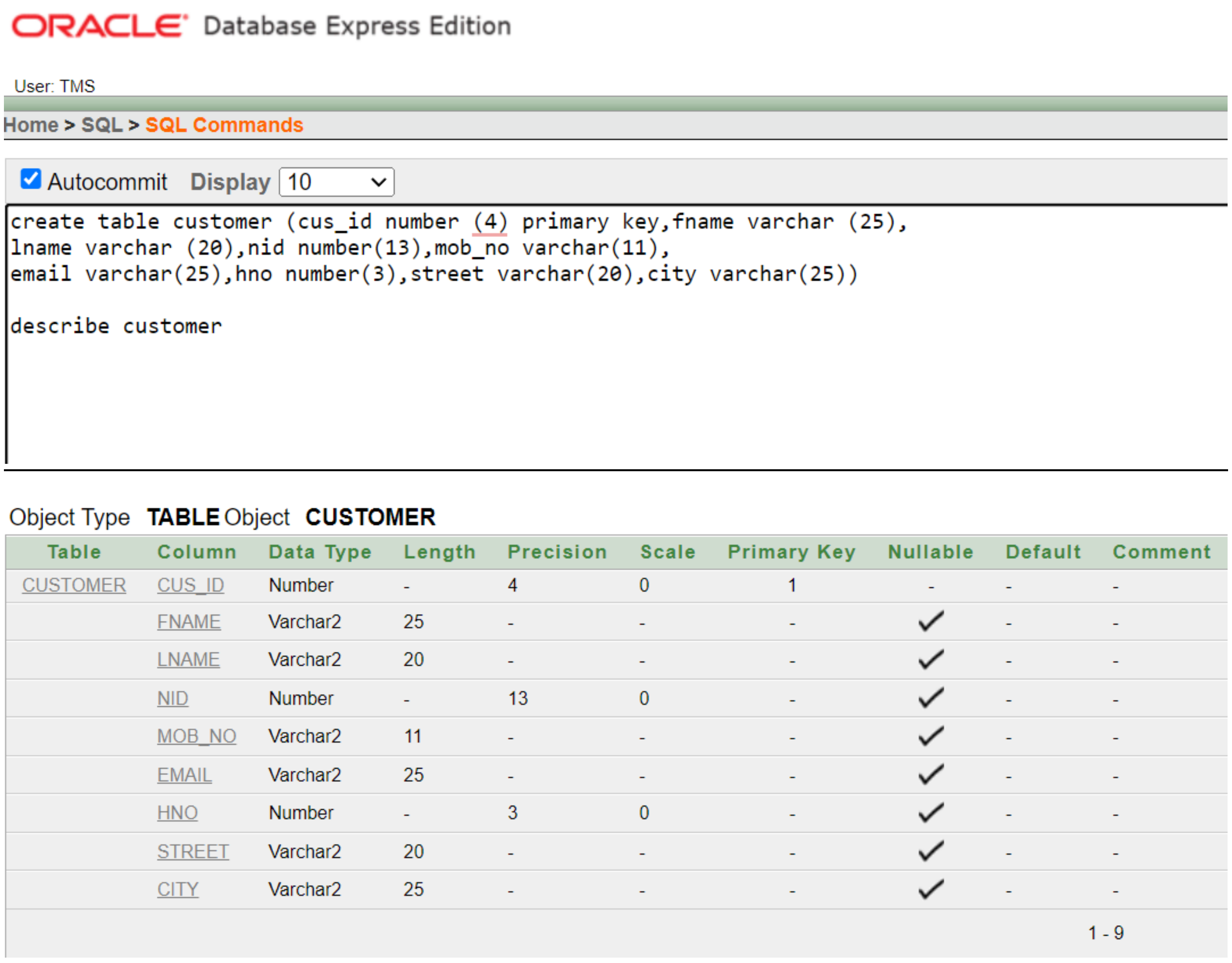
5. Finalization:

1. cus_id (PK), fname, lname, nid, hno, city, street, mob_no, email
2. h_id (PK), h_name, h_loc, h_contact, h_license,h_cost
3. cus_id (PK), h_id (FK),r_amm
4. t_no(PK), dept_loc, arr_loc, dept_time, arr_time, t_cost
5. ~~cus_id (PK), fname, lname, nid, hno, city, street, mob_no, email~~ ×
6. cus_id (PK), t_no(FK),s_amm
7. p_id(PK), p_name, destination, p_cost
8. cus_id (PK), p_id(FK),b_amm
9. ~~cus_id (PK), fname, lname, nid, hno, city, street, mob_no, email~~ ×
10. ~~cus_id (PK), fname, lname, nid, hno, city, street, mob_no, email~~ ×
11. pay_id(PK), amount, co_name, c_no, cvc, expire
12. cus_id (PK), pay_id (FK)
13. ~~cus_id (PK), fname, lname, nid, hno, city, street, mob_no, email~~ ×
14. rev_id(PK), rating, r_date, comment
15. cus_id (PK), rev_id (FK)

Final Tables:

1. cus_id (PK), fname, lname, nid, hno, city, street, mob_no, email (Customer)
2. h_id (PK), h_name, h_loc, h_contact, h_license, h_cost (Hotel)
3. cus_id (PK), h_id (FK),r_amm (Reserves)
4. t_no(PK), dept_loc, arr_loc, dept_time, arr_time, t_cost (Transport)
5. cus_id (PK), t_no(FK),s_amm (Schedules)
6. p_id(PK), p_name, destination, p_cost (Package)
7. cus_id (PK), p_id(FK),b_amm (Books)
8. pay_id(PK), amount, co_name, c_no, cvc, expire (Payment)
9. cus_id (PK), pay_id (FK) (Makes)
10. rev_id(PK), rating, r_date, comment (Review)
11. cus_id (PK), rev_id (FK) (Gives)

6.Table Creation:


ORACLE Database Express Edition

User: TMS

Home > SQL > SQL Commands

Autocommit ✓ Display 10

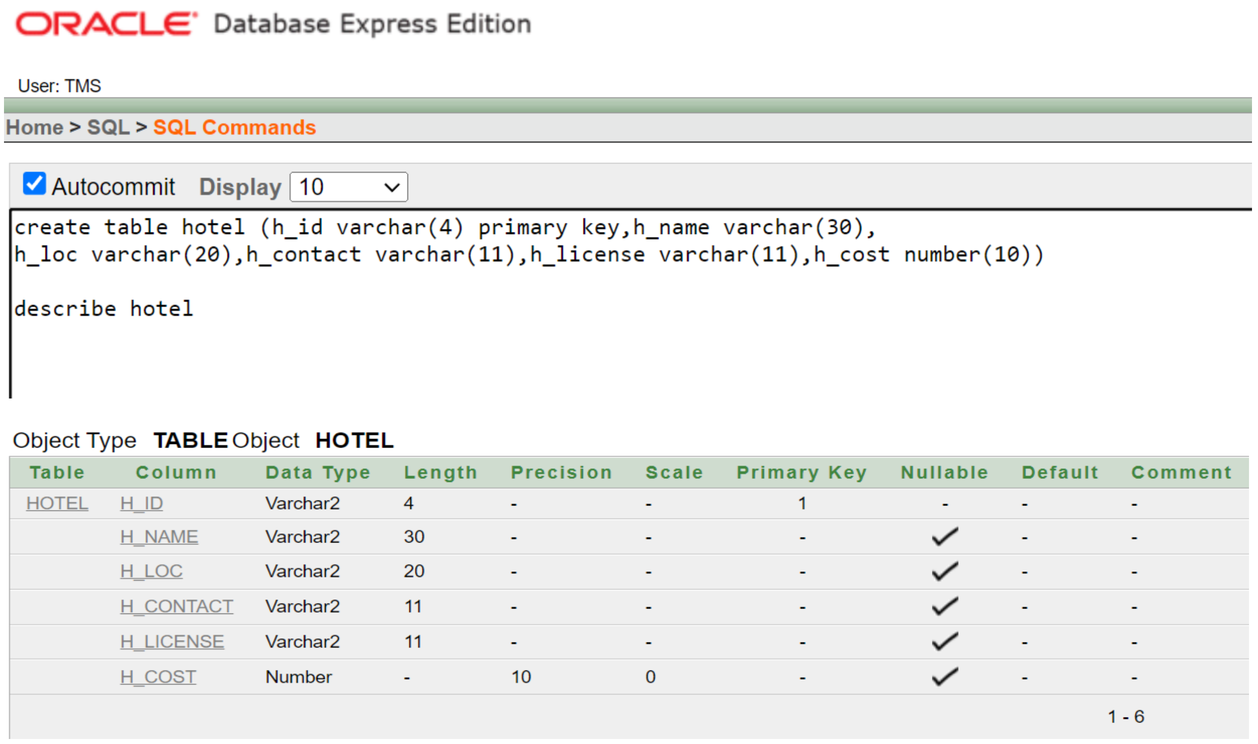
```
create table customer (cus_id number (4) primary key,fname varchar (25),
lname varchar (20),nid number(13),mob_no varchar(11),
email varchar(25),hno number(3),street varchar(20),city varchar(25))

describe customer
```

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

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSTOMER	CUS_ID	Number	-	4	0	1	-	-	-
	FNAME	Varchar2	25	-	-	-	✓	-	-
	LNAME	Varchar2	20	-	-	-	✓	-	-
	NID	Number	-	13	0	-	✓	-	-
	MOB_NO	Varchar2	11	-	-	-	✓	-	-
	EMAIL	Varchar2	25	-	-	-	✓	-	-
	HNO	Number	-	3	0	-	✓	-	-
	STREET	Varchar2	20	-	-	-	✓	-	-
	CITY	Varchar2	25	-	-	-	✓	-	-

1 - 9

Figure 6.1: Command & Table of Customer



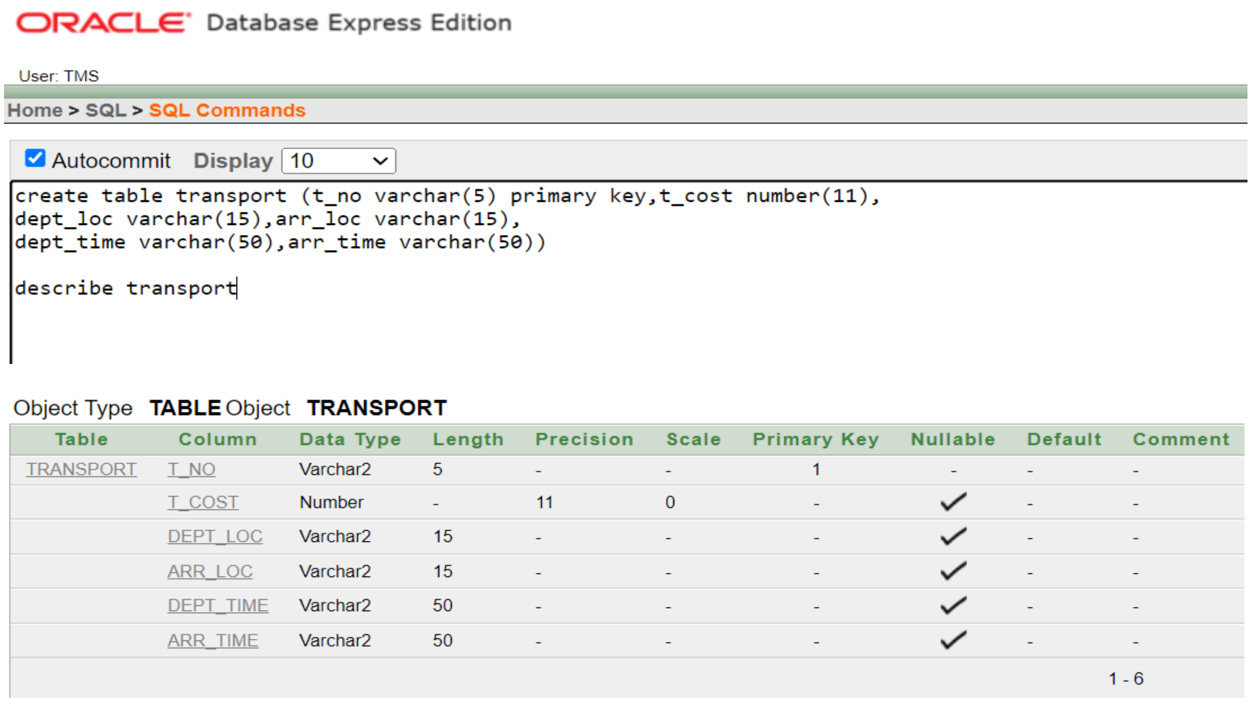
```
create table hotel (h_id varchar(4) primary key,h_name varchar(30),
h_loc varchar(20),h_contact varchar(11),h_license varchar(11),h_cost number(10))

describe hotel
```

Object Type **TABLE** Object **HOTEL**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
HOTEL	H_ID	Varchar2	4	-	-	1	-	-	-
	H_NAME	Varchar2	30	-	-	-	✓	-	-
	H_LOC	Varchar2	20	-	-	-	✓	-	-
	H_CONTACT	Varchar2	11	-	-	-	✓	-	-
	H_LICENSE	Varchar2	11	-	-	-	✓	-	-
	H_COST	Number	-	10	0	-	✓	-	-

1 - 6

Figure 6.2: Command & Table of Hotel



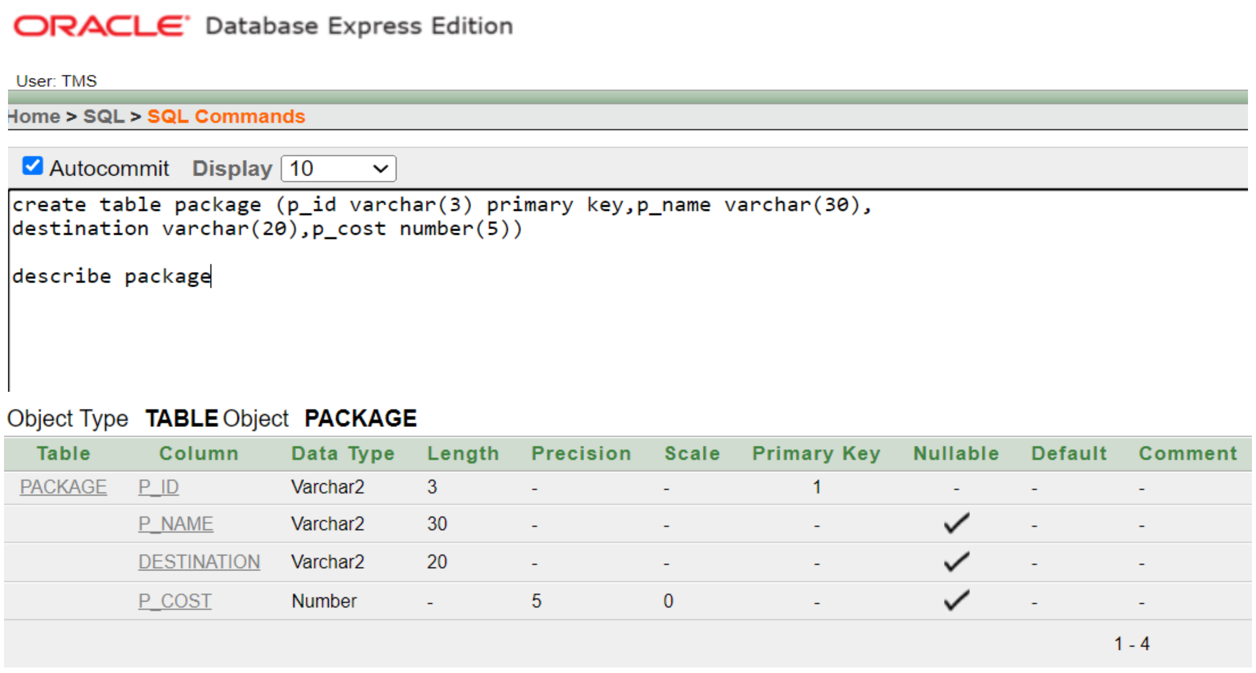
```
create table transport (t_no varchar(5) primary key,t_cost number(11),
dept_loc varchar(15),arr_loc varchar(15),
dept_time varchar(50),arr_time varchar(50))

describe transport
```

Object Type **TABLE** Object **TRANSPORT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TRANSPORT	T_NO	Varchar2	5	-	-	1	-	-	-
	T_COST	Number	-	11	0	-	✓	-	-
	DEPT_LOC	Varchar2	15	-	-	-	✓	-	-
	ARR_LOC	Varchar2	15	-	-	-	✓	-	-
	DEPT_TIME	Varchar2	50	-	-	-	✓	-	-
	ARR_TIME	Varchar2	50	-	-	-	✓	-	-

1 - 6

Figure 6.3: Command & Table of Transport



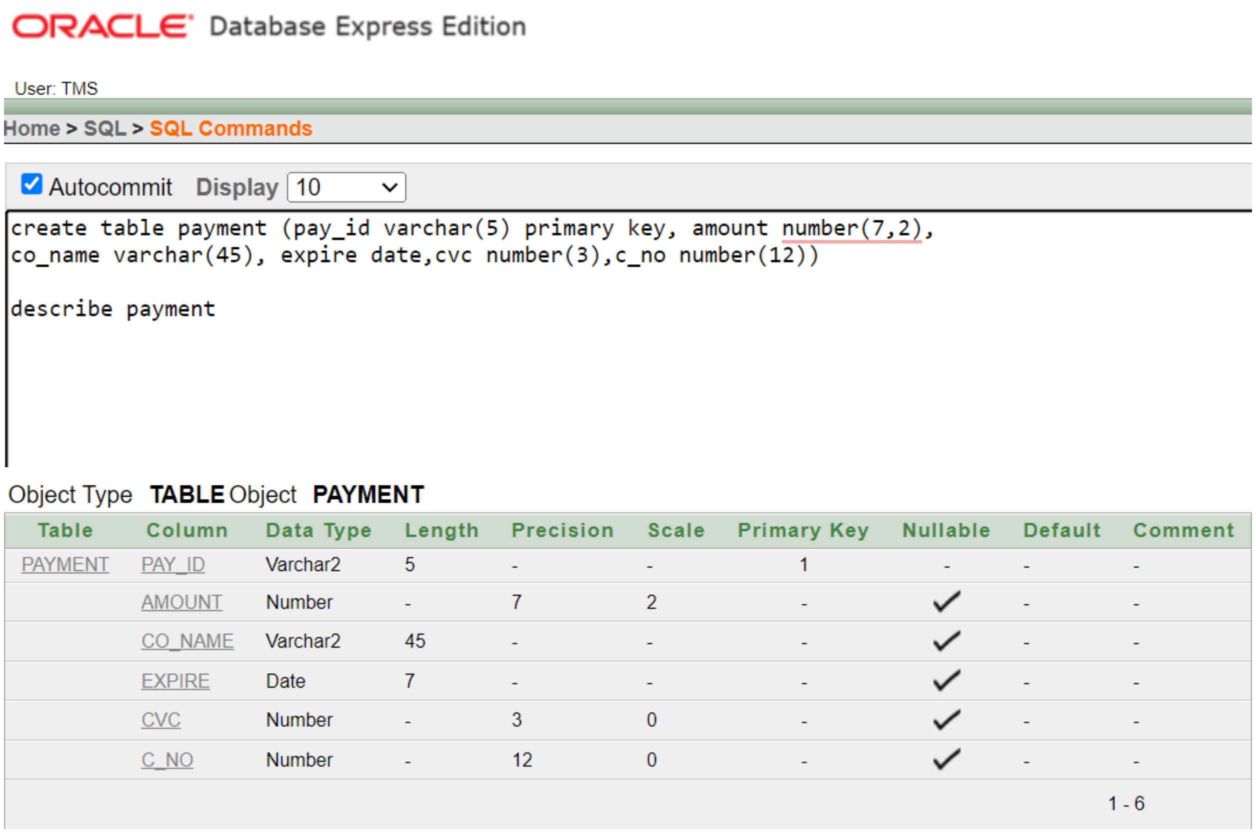
```
create table package (p_id varchar(3) primary key,p_name varchar(30),
destination varchar(20),p_cost number(5))

describe package
```

Object Type **TABLE** Object **PACKAGE**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PACKAGE	P_ID	Varchar2	3	-	-	1	-	-	-
	P_NAME	Varchar2	30	-	-	-	✓	-	-
	DESTINATION	Varchar2	20	-	-	-	✓	-	-
	P_COST	Number	-	5	0	-	✓	-	-

1 - 4

Figure 6.4: Command & Table of Package



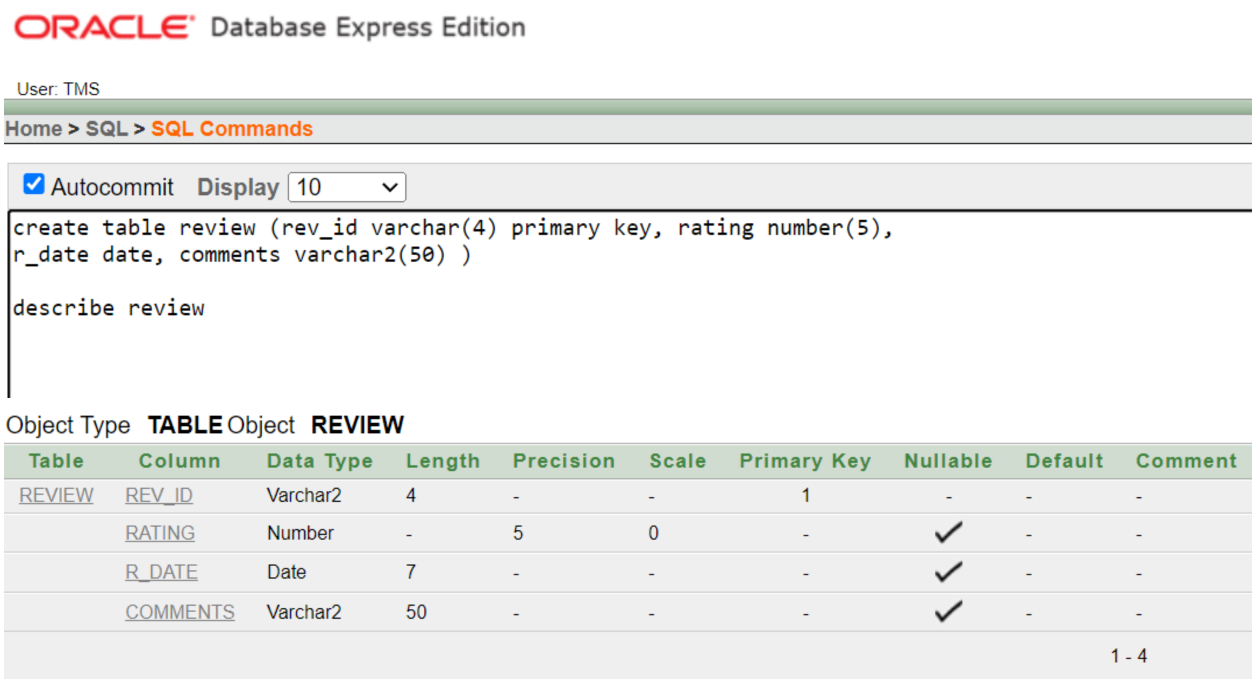
```
create table payment (pay_id varchar(5) primary key, amount number(7,2),
co_name varchar(45), expire date,cvc number(3),c_no number(12))

describe payment
```

Object Type **TABLE** Object **PAYMENT**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PAYMENT	PAY_ID	Varchar2	5	-	-	1	-	-	-
	AMOUNT	Number	-	7	2	-	✓	-	-
	CO_NAME	Varchar2	45	-	-	-	✓	-	-
	EXPIRE	Date	7	-	-	-	✓	-	-
	CVC	Number	-	3	0	-	✓	-	-
	C_NO	Number	-	12	0	-	✓	-	-

1 - 6

Figure 6.5: Command & Table of Payment



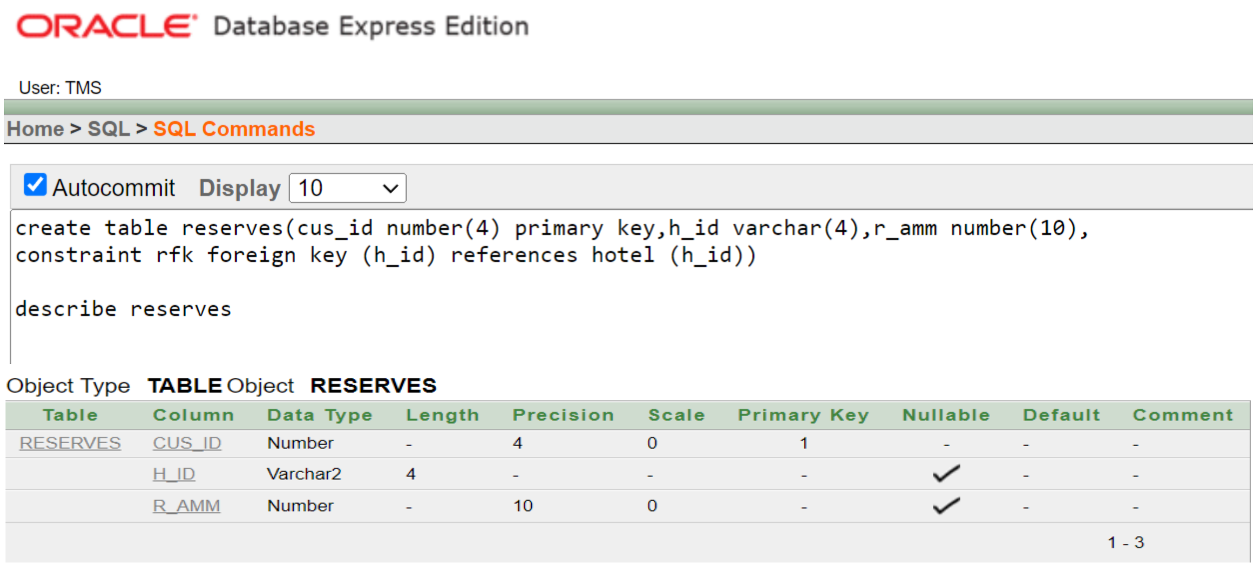
```
create table review (rev_id varchar(4) primary key, rating number(5),
r_date date, comments varchar2(50) )

describe review
```

Object Type **TABLE** Object **REVIEW**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
REVIEW	REV_ID	Varchar2	4	-	-	1	-	-	-
	RATING	Number	-	5	0	-	✓	-	-
	R_DATE	Date	7	-	-	-	✓	-	-
	COMMENTS	Varchar2	50	-	-	-	✓	-	-

1 - 4

Figure 6.6: Command & Table of Review



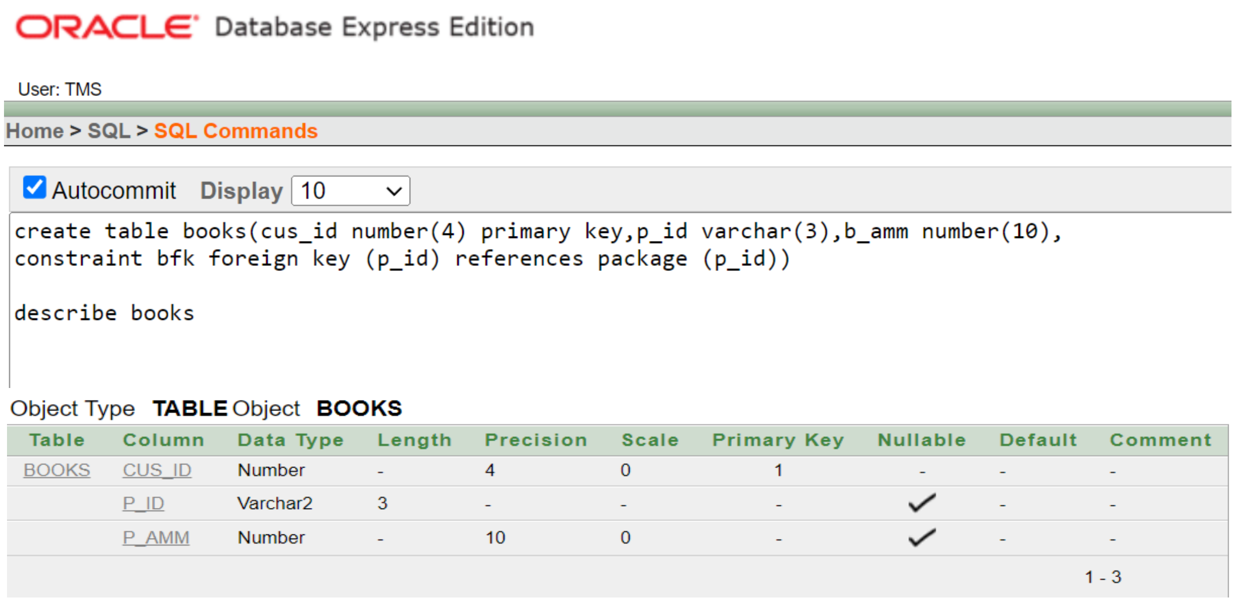
```
create table reserves(cus_id number(4) primary key,h_id varchar(4),r_amm number(10),
constraint rfk foreign key (h_id) references hotel (h_id))

describe reserves
```

Object Type **TABLE** Object **RESERVES**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESERVES	CUS_ID	Number	-	4	0	1	-	-	-
	H_ID	Varchar2	4	-	-	-	✓	-	-
	R_AMM	Number	-	10	0	-	✓	-	-

1 - 3

Figure 6.7: Command & Table of Reserves



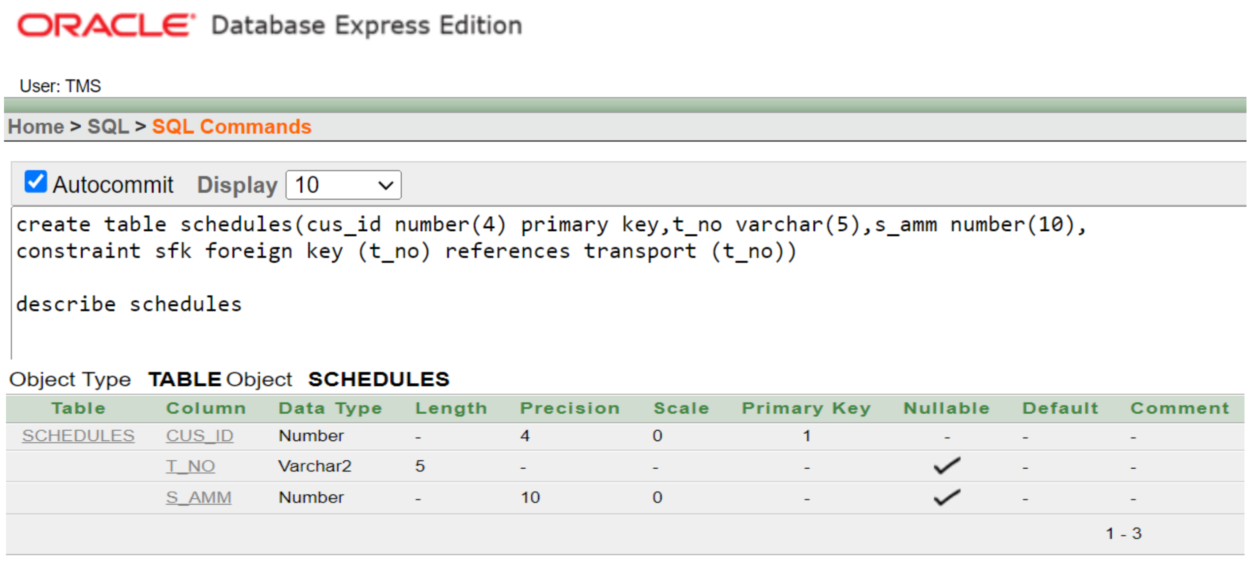
```
create table books(cus_id number(4) primary key,p_id varchar(3),b_amm number(10),
constraint bfk foreign key (p_id) references package (p_id))

describe books
```

Object Type **TABLE** Object **BOOKS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BOOKS	CUS_ID	Number	-	4	0	1	-	-	-
	P_ID	Varchar2	3	-	-	-	✓	-	-
	P_AMM	Number	-	10	0	-	✓	-	-

1 - 3

Figure 6.8: Command & Table of Books



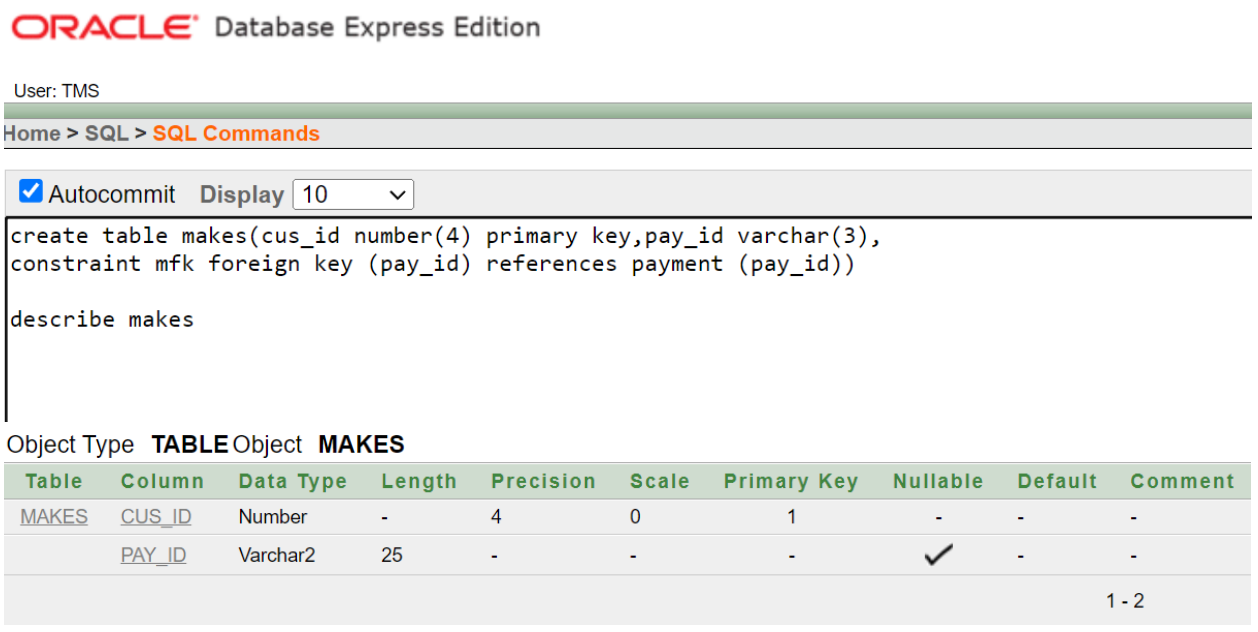
```
create table schedules(cus_id number(4) primary key,t_no varchar(5),s_amm number(10),
constraint sfk foreign key (t_no) references transport (t_no))

describe schedules
```

Object Type **TABLE** Object **SCHEDULES**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SCHEDULES	CUS_ID	Number	-	4	0	1	-	-	-
	T_NO	Varchar2	5	-	-	-	✓	-	-
	S_AMM	Number	-	10	0	-	✓	-	-

1 - 3

Figure 6.9: Command & Table of Schedules



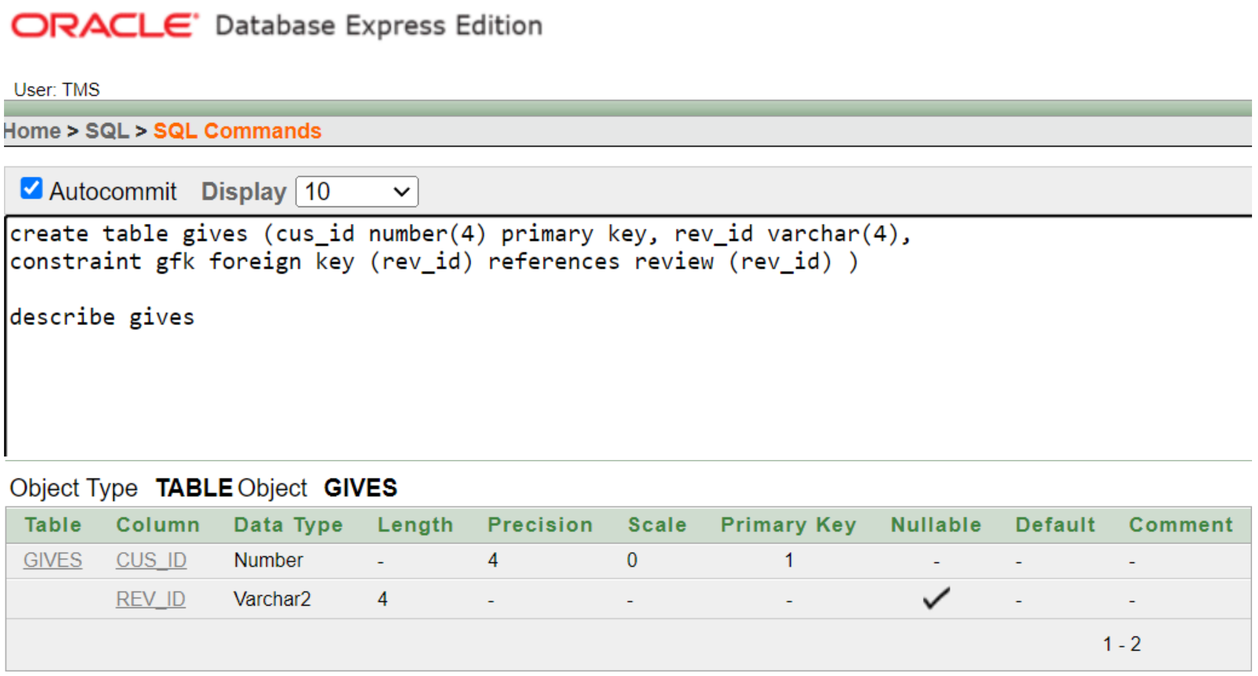
```
create table makes(cus_id number(4) primary key,pay_id varchar(3),
constraint mfk foreign key (pay_id) references payment (pay_id))

describe makes
```

Object Type **TABLE** Object **MAKES**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MAKES	CUS_ID	Number	-	4	0	1	-	-	-
	PAY_ID	Varchar2	25	-	-	-	✓	-	-

1 - 2

Figure 6.10: Command & Table of Makes



```
create table gives (cus_id number(4) primary key, rev_id varchar(4),
constraint gfk foreign key (rev_id) references review (rev_id) )

describe gives
```

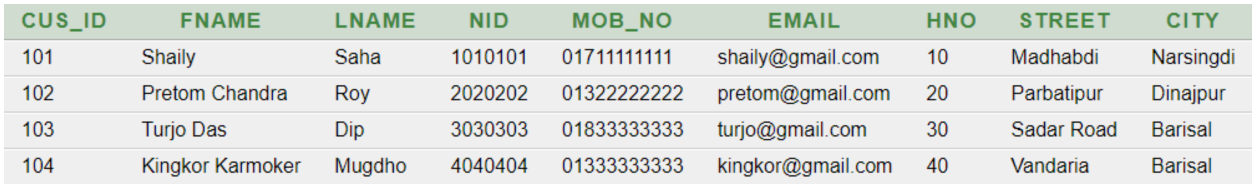
Object Type **TABLE** Object **GIVES**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
GIVES	CUS_ID	Number	-	4	0	1	-	-	-
	REV_ID	Varchar2	4	-	-	-	✓	-	-

1 - 2

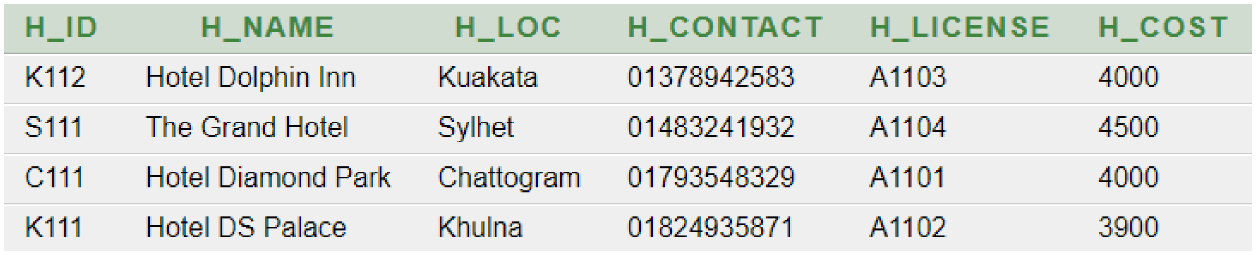
Figure 6.11: Command & Table of Gives

7.DATA INSERTION:



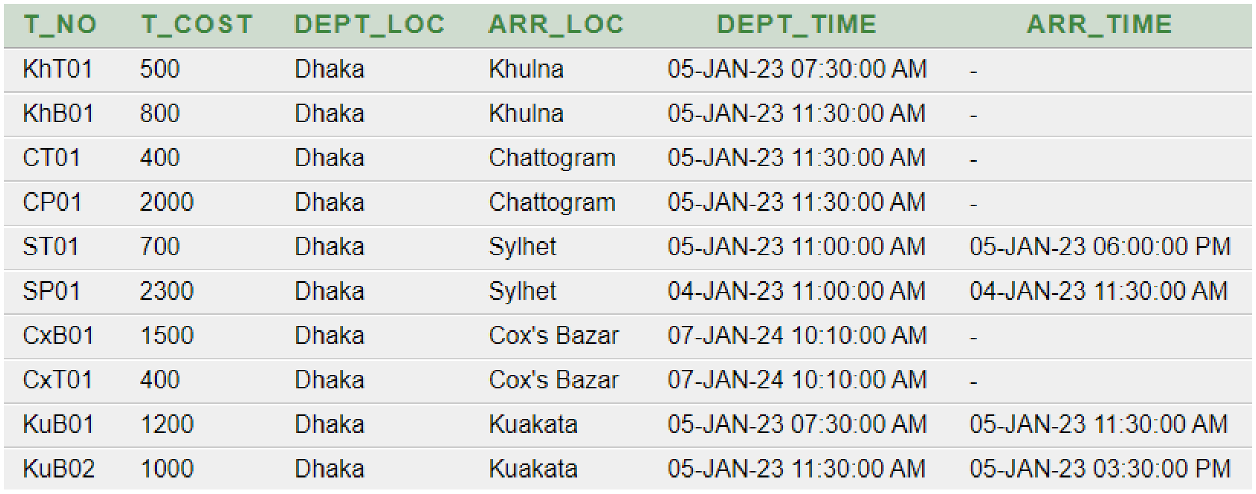
CUS_ID	FNAME	LNAME	NID	MOB_NO	EMAIL	HNO	STREET	CITY
101	Shaily	Saha	1010101	01711111111	shaily@gmail.com	10	Madhabdi	Narsingdi
102	Pretom Chandra	Roy	2020202	01322222222	pretom@gmail.com	20	Parbatipur	Dinajpur
103	Turjo Das	Dip	3030303	01833333333	turjo@gmail.com	30	Sadar Road	Barisal
104	Kingkor Karmoker	Mugdho	4040404	01333333333	kingkor@gmail.com	40	Vandaria	Barisal

Figure 7.1 : Customer Table



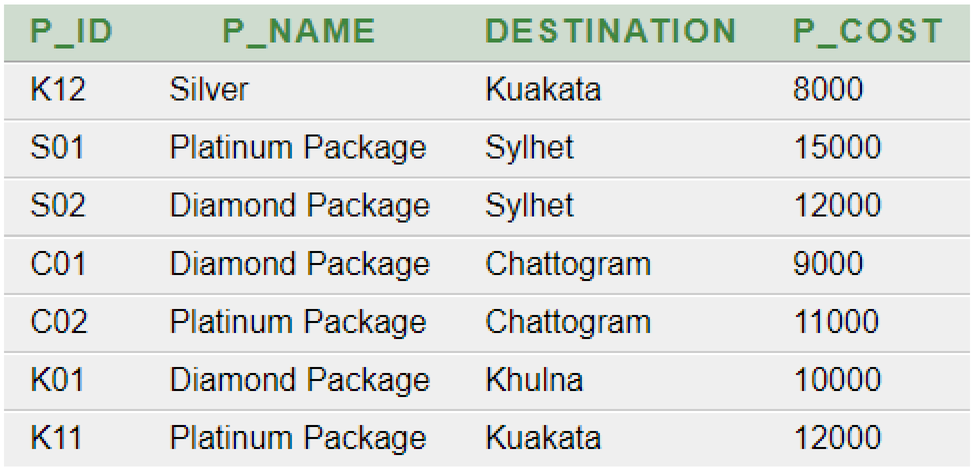
H_ID	H_NAME	H_LOC	H_CONTACT	H_LICENSE	H_COST
K112	Hotel Dolphin Inn	Kuakata	01378942583	A1103	4000
S111	The Grand Hotel	Sylhet	01483241932	A1104	4500
C111	Hotel Diamond Park	Chattogram	01793548329	A1101	4000
K111	Hotel DS Palace	Khulna	01824935871	A1102	3900

Figure 7.2 : Hotel Table



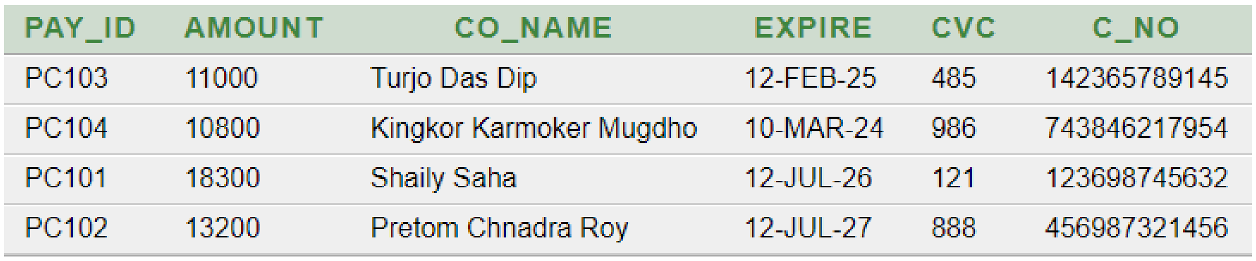
T_NO	T_COST	DEPT_LOC	ARR_LOC	DEPT_TIME	ARR_TIME
KhT01	500	Dhaka	Khulna	05-JAN-23 07:30:00 AM	-
KhB01	800	Dhaka	Khulna	05-JAN-23 11:30:00 AM	-
CT01	400	Dhaka	Chattogram	05-JAN-23 11:30:00 AM	-
CP01	2000	Dhaka	Chattogram	05-JAN-23 11:30:00 AM	-
ST01	700	Dhaka	Sylhet	05-JAN-23 11:00:00 AM	05-JAN-23 06:00:00 PM
SP01	2300	Dhaka	Sylhet	04-JAN-23 11:00:00 AM	04-JAN-23 11:30:00 AM
CxB01	1500	Dhaka	Cox's Bazar	07-JAN-24 10:10:00 AM	-
CxT01	400	Dhaka	Cox's Bazar	07-JAN-24 10:10:00 AM	-
KuB01	1200	Dhaka	Kuakata	05-JAN-23 07:30:00 AM	05-JAN-23 11:30:00 AM
KuB02	1000	Dhaka	Kuakata	05-JAN-23 11:30:00 AM	05-JAN-23 03:30:00 PM

Figure 7.3 : Transport Table



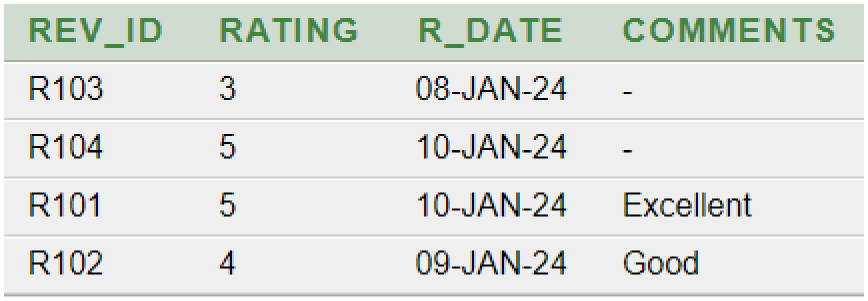
P_ID	P_NAME	DESTINATION	P_COST
K12	Silver	Kuakata	8000
S01	Platinum Package	Sylhet	15000
S02	Diamond Package	Sylhet	12000
C01	Diamond Package	Chattogram	9000
C02	Platinum Package	Chattogram	11000
K01	Diamond Package	Khulna	10000
K11	Platinum Package	Kuakata	12000

Figure 7.4 : Package Table



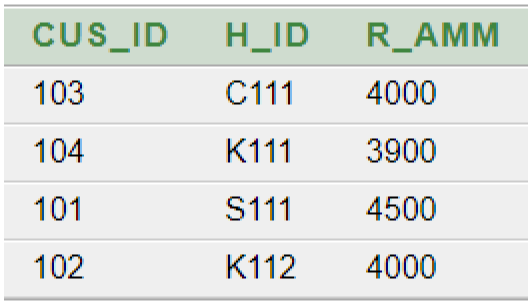
PAY_ID	AMOUNT	CO_NAME	EXPIRE	CVC	C_NO
PC103	11000	Turjo Das Dip	12-FEB-25	485	142365789145
PC104	10800	Kingkor Karmoker Mugdho	10-MAR-24	986	743846217954
PC101	18300	Shaily Saha	12-JUL-26	121	123698745632
PC102	13200	Pretom Chnadra Roy	12-JUL-27	888	456987321456

Figure 7.5 : Payment Table



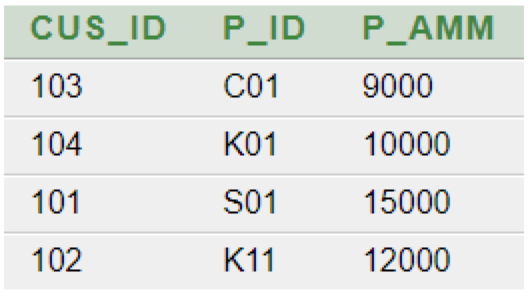
REV_ID	RATING	R_DATE	COMMENTS
R103	3	08-JAN-24	-
R104	5	10-JAN-24	-
R101	5	10-JAN-24	Excellent
R102	4	09-JAN-24	Good

Figure 7.6 : Review Table



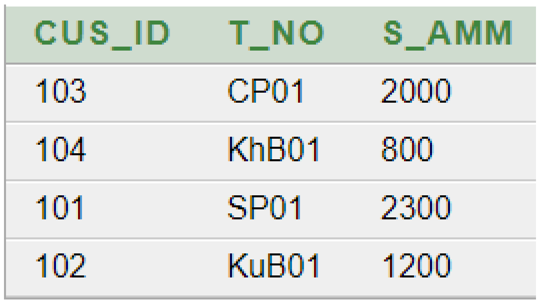
CUS_ID	H_ID	R_AMM
103	C111	4000
104	K111	3900
101	S111	4500
102	K112	4000

Figure 7.7 : Reserves Table



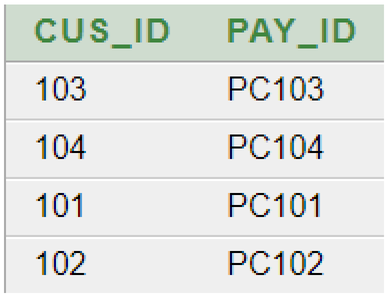
CUS_ID	P_ID	P_AMM
103	C01	9000
104	K01	10000
101	S01	15000
102	K11	12000

Figure 7.8 : Books Table



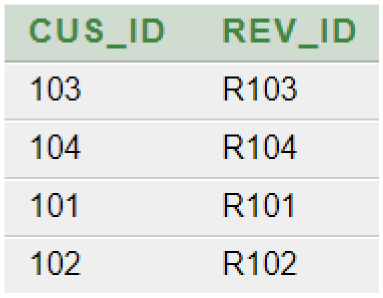
CUS_ID	T_NO	S_AMM
103	CP01	2000
104	KhB01	800
101	SP01	2300
102	KuB01	1200

Figure 7.9 : Schedules Table



CUS_ID	PAY_ID
103	PC103
104	PC104
101	PC101
102	PC102

Figure 7.10 : Makes Table

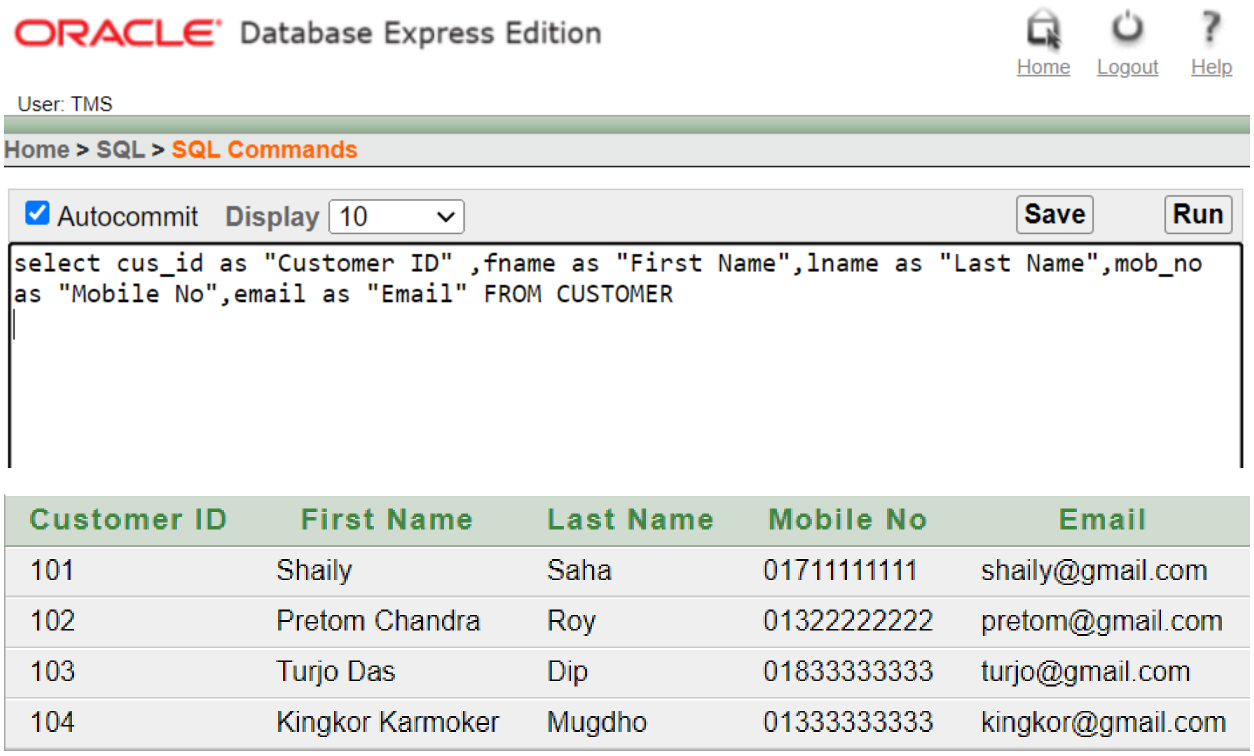


CUS_ID	REV_ID
103	R103
104	R104
101	R101
102	R102

Figure 7.11 : Gives Table

8.Query Test:

A. Simple Query:
Show the Customer ID, First Name , Last Name , Mobile Number , Email from customer table.



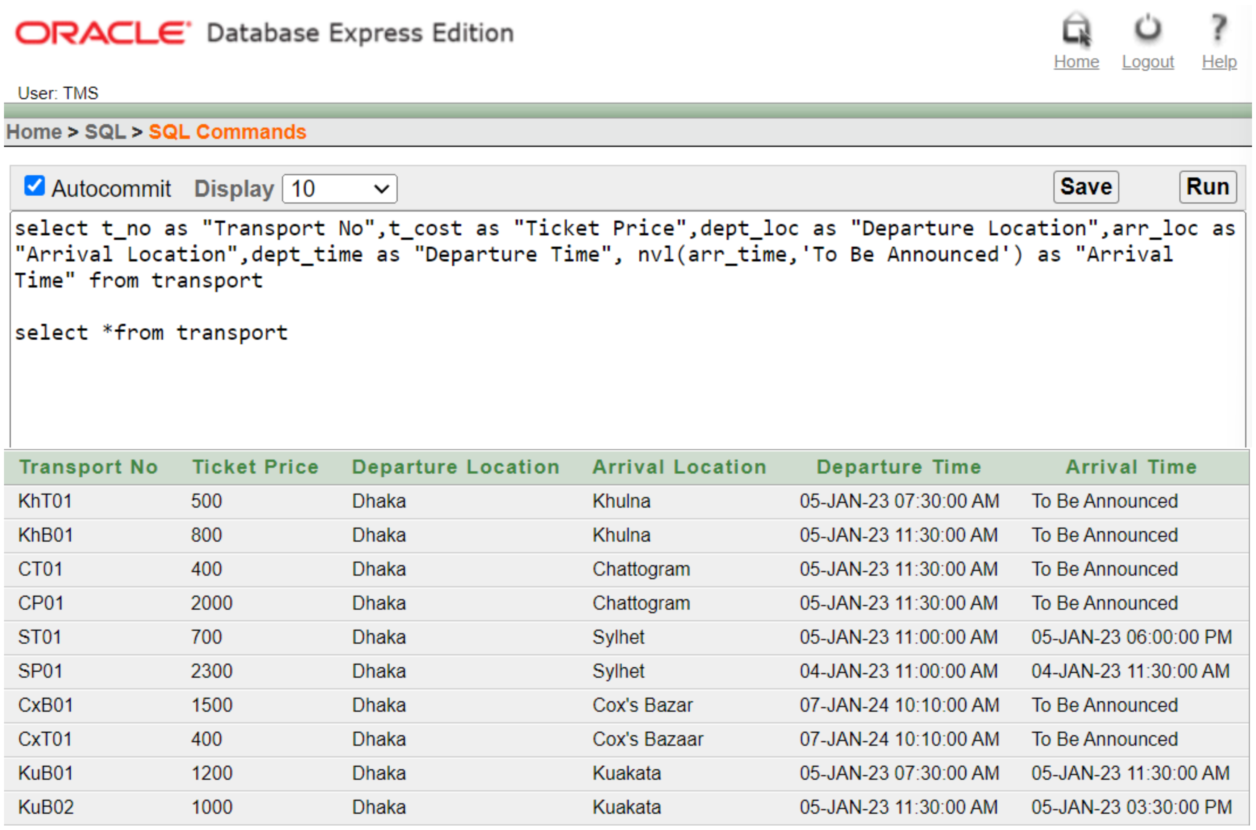
```
select cus_id as "Customer ID" ,fname as "First Name",lname as "Last Name",mob_no
as "Mobile No",email as "Email" FROM CUSTOMER
```

Customer ID	First Name	Last Name	Mobile No	Email
101	Shaily	Saha	01711111111	shaily@gmail.com
102	Pretom Chandra	Roy	01322222222	pretom@gmail.com
103	Turjo Das	Dip	01833333333	turjo@gmail.com
104	Kingkor Karmoker	Mugdho	01333333333	kingkor@gmail.com

Fig: Simple Query with Result

B. Query with A Single Row Function:

Show all data from transport and replace the null value of the arrival time with the string “To Be Announced”.



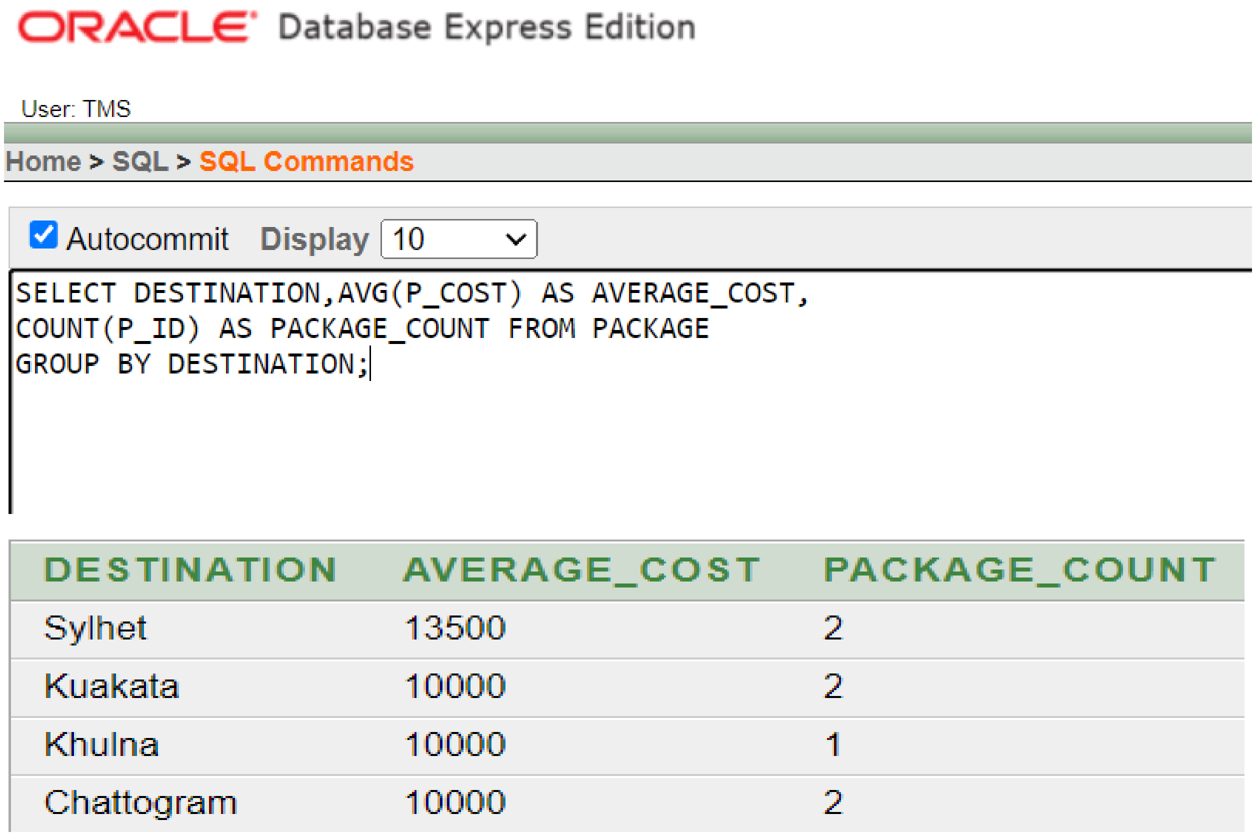
```
select t_no as "Transport No",t_cost as "Ticket Price",dept_loc as "Departure Location",arr_loc as
"Arrival Location",dept_time as "Departure Time", nvl(arr_time,'To Be Announced') as "Arrival
Time" from transport

select *from transport
```

Transport No	Ticket Price	Departure Location	Arrival Location	Departure Time	Arrival Time
KhT01	500	Dhaka	Khulna	05-JAN-23 07:30:00 AM	To Be Announced
KhB01	800	Dhaka	Khulna	05-JAN-23 11:30:00 AM	To Be Announced
CT01	400	Dhaka	Chattogram	05-JAN-23 11:30:00 AM	To Be Announced
CP01	2000	Dhaka	Chattogram	05-JAN-23 11:30:00 AM	To Be Announced
ST01	700	Dhaka	Sylhet	05-JAN-23 11:00:00 AM	05-JAN-23 06:00:00 PM
SP01	2300	Dhaka	Sylhet	04-JAN-23 11:00:00 AM	04-JAN-23 11:30:00 AM
CxB01	1500	Dhaka	Cox's Bazar	07-JAN-24 10:10:00 AM	To Be Announced
CxT01	400	Dhaka	Cox's Bazaar	07-JAN-24 10:10:00 AM	To Be Announced
KuB01	1200	Dhaka	Kuakata	05-JAN-23 07:30:00 AM	05-JAN-23 11:30:00 AM
KuB02	1000	Dhaka	Kuakata	05-JAN-23 11:30:00 AM	05-JAN-23 03:30:00 PM

Fig: Query with A Single Row Function

C. Query with A Multiple Row Function / Aggregate Function:

Show the average cost and total count of packages for each destination in the package table.



```
SELECT DESTINATION,AVG(P_COST) AS AVERAGE_COST,
COUNT(P_ID) AS PACKAGE_COUNT FROM PACKAGE
GROUP BY DESTINATION;
```

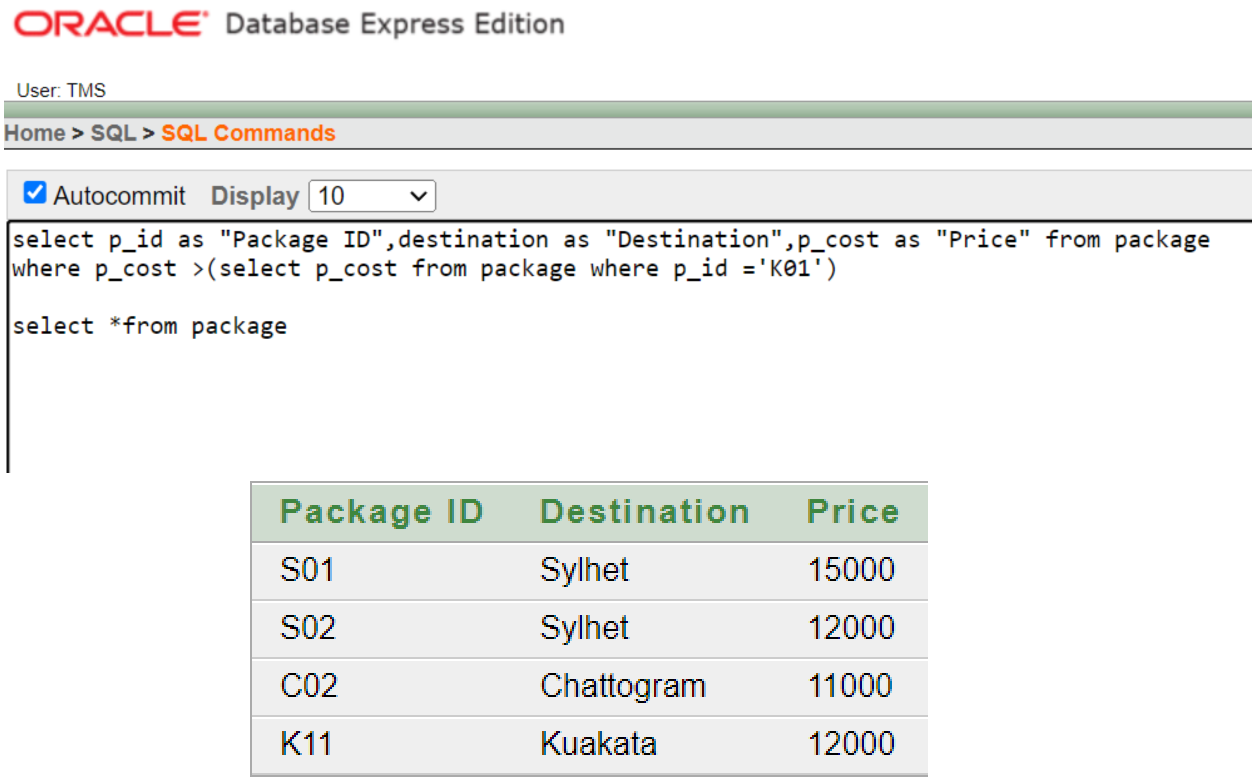
DESTINATION	AVERAGE_COST	PACKAGE_COUNT
Sylhet	13500	2
Kuakata	10000	2
Khulna	10000	1
Chattogram	10000	2

Fig: Query with a Multiple row function/ Aggregate function with result

D. Single Row Subquery and Multiple Row Subquery:

Single Row Subquery:

Show the package IDs, destinations, and costs of packages that have a cost higher than the package with ID 'K01'



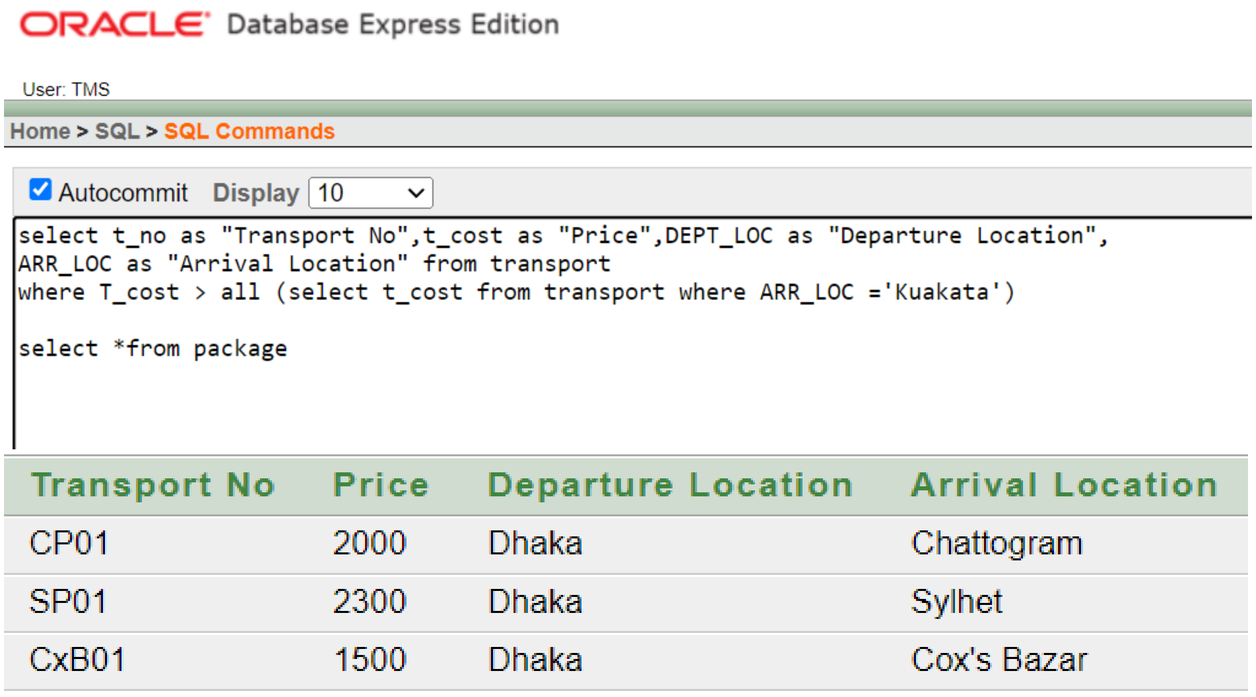
```
select p_id as "Package ID",destination as "Destination",p_cost as "Price" from package
where p_cost >(select p_cost from package where p_id ='K01')

select *from package
```

Package ID	Destination	Price
S01	Sylhet	15000
S02	Sylhet	12000
C02	Chattogram	11000
K11	Kuakata	12000

Fig: Single Row Sub Query with Result

Multiple Row Subquery:

Show transport numbers, costs, departure locations (DEPT_LOC), and arrival locations (ARR_LOC) for transports with costs greater than all the transports arriving at 'Kuakata'.



```
select t_no as "Transport No",t_cost as "Price",DEPT_LOC as "Departure Location",
ARR_LOC as "Arrival Location" from transport
where T_cost > all (select t_cost from transport where ARR_LOC ='Kuakata')

select *from package
```

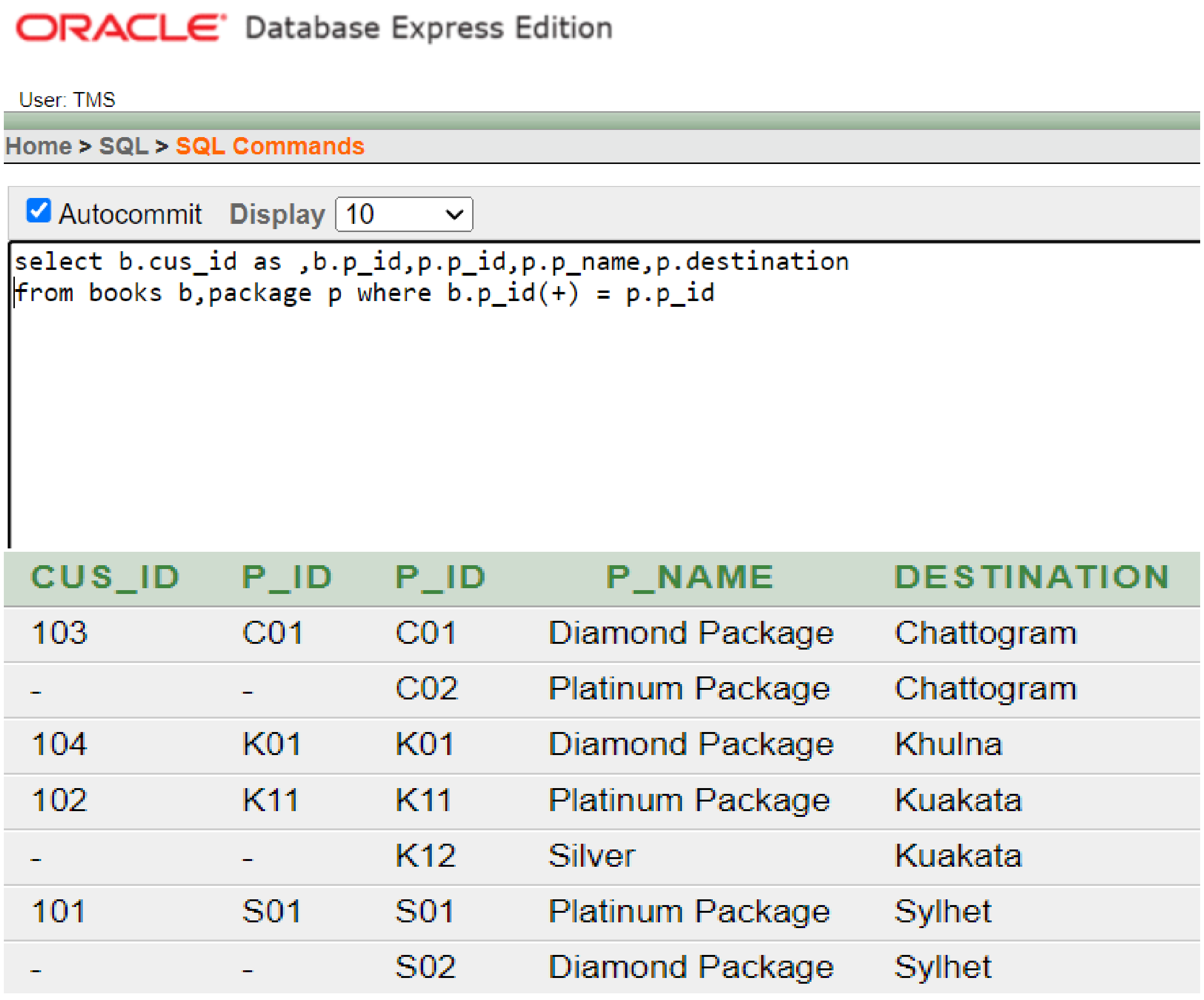
Transport No	Price	Departure Location	Arrival Location
CP01	2000	Dhaka	Chattogram
SP01	2300	Dhaka	Sylhet
CxB01	1500	Dhaka	Cox's Bazar

Fig: Multiple Row Sub Query with Result

E. Any 2 Kinds of Joining:

Outer Joining:

Perform an outer join with the Books and Package table, showing the customer IDs, package IDs, package names, and destinations.



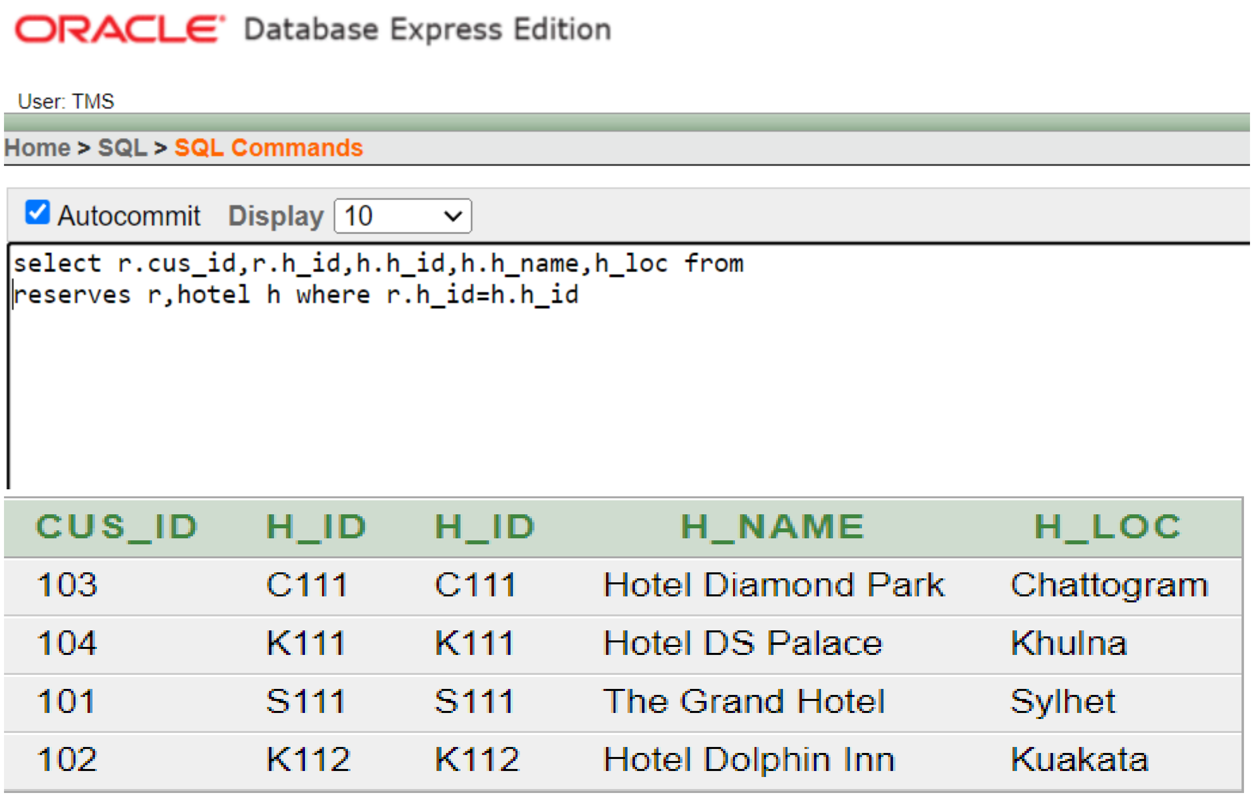
```
select b.cus_id as ,b.p_id,p.p_id,p.p_name,p.destination
from books b,package p where b.p_id(+) = p.p_id
```

CUS_ID	P_ID	P_ID	P_NAME	DESTINATION
103	C01	C01	Diamond Package	Chattogram
-	-	C02	Platinum Package	Chattogram
104	K01	K01	Diamond Package	Khulna
102	K11	K11	Platinum Package	Kuakata
-	-	K12	Silver	Kuakata
101	S01	S01	Platinum Package	Sylhet
-	-	S02	Diamond Package	Sylhet

Fig: Outer Join with Result

Equi Joining:

Perform an equi join with the hotel and reserve table, showing the customer IDs, Hotel IDs, Hotel names, and Hotel location.



```
select r.cus_id,r.h_id,h.h_id,h.h_name,h_loc from
reserves r,hotel h where r.h_id=h.h_id
```

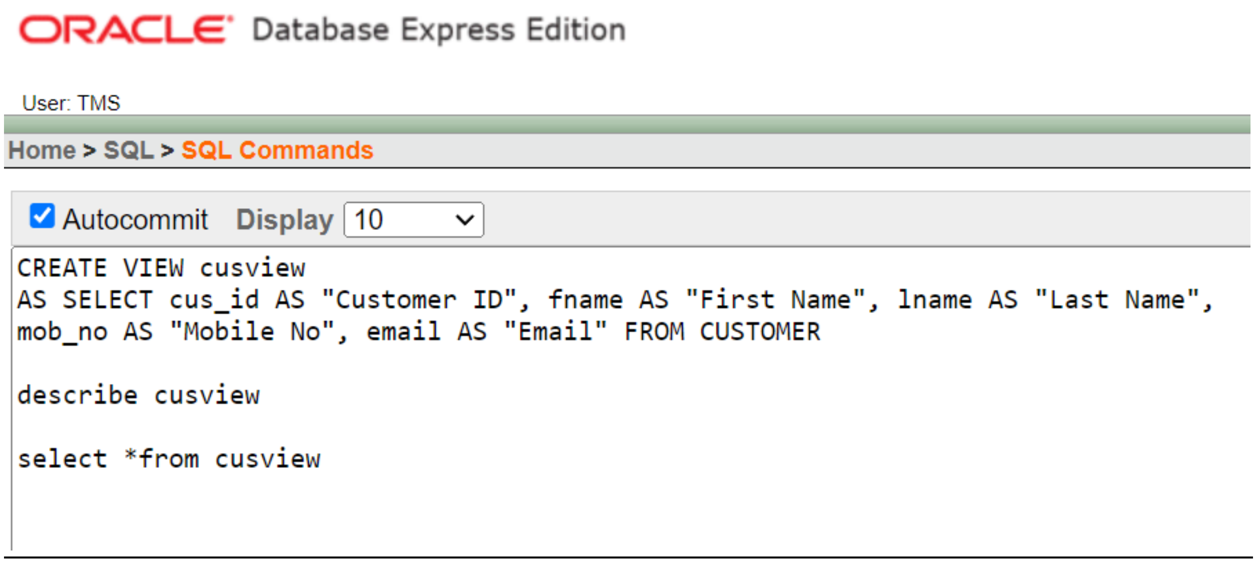
CUS_ID	H_ID	H_ID	H_NAME	H_LOC
103	C111	C111	Hotel Diamond Park	Chattogram
104	K111	K111	Hotel DS Palace	Khulna
101	S111	S111	The Grand Hotel	Sylhet
102	K112	K112	Hotel Dolphin Inn	Kuakata

Fig: Equi Join with Result

F. View:

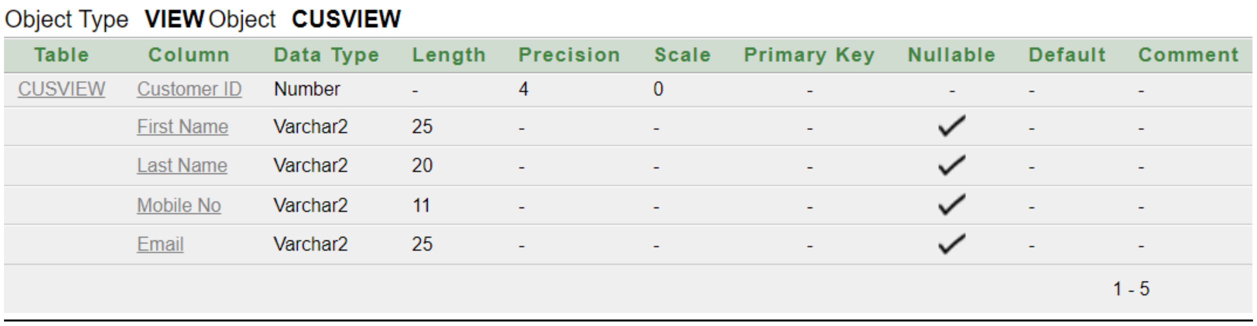
Simple View:

Create a simple view showing the Customer ID, First Name, Last Name, Mobile Number, Email from customer table.



```
CREATE VIEW cusview
AS SELECT cus_id AS "Customer ID", fname AS "First Name", lname AS "Last Name",
mob_no AS "Mobile No", email AS "Email" FROM CUSTOMER

describe cusview

select *from cusview
```

Fig F.1: Command Of Simple View

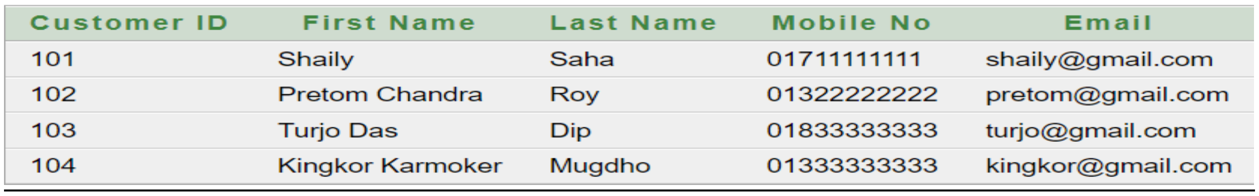


Object Type **VIEW** Object **CUSVIEW**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
CUSVIEW	Customer ID	Number	-	4	0	-	-	-	-
	First Name	Varchar2	25	-	-	-	✓	-	-
	Last Name	Varchar2	20	-	-	-	✓	-	-
	Mobile No	Varchar2	11	-	-	-	✓	-	-
	Email	Varchar2	25	-	-	-	✓	-	-

1 - 5

Fig F.2: Description Of Simple View

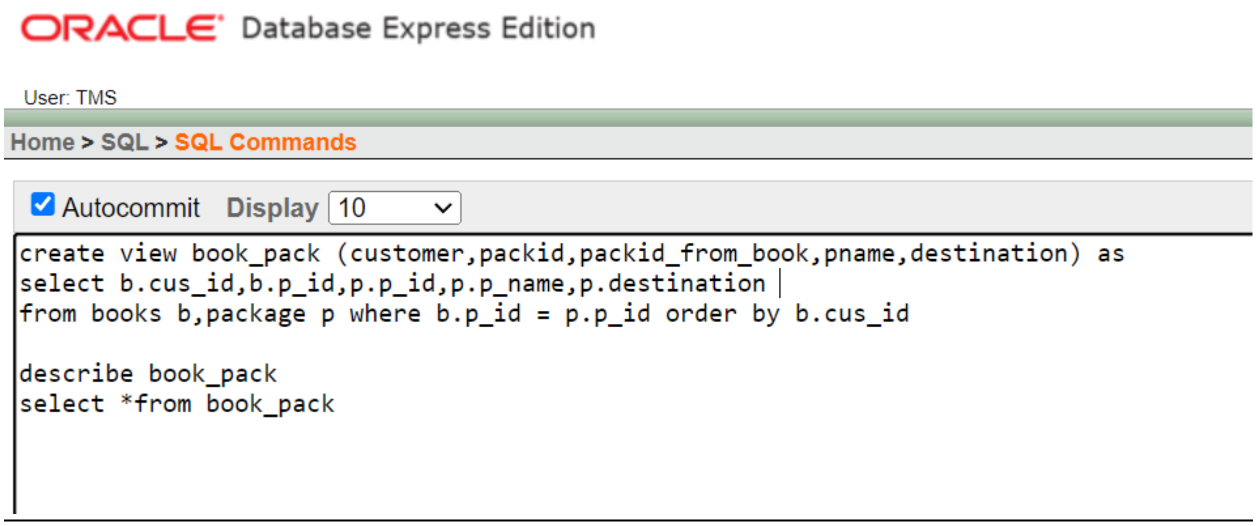


Customer ID	First Name	Last Name	Mobile No	Email
101	Shaily	Saha	01711111111	shaily@gmail.com
102	Pretom Chandra	Roy	01322222222	pretom@gmail.com
103	Turjo Das	Dip	01833333333	turjo@gmail.com
104	Kingkor Karmoker	Mugdho	01333333333	kingkor@gmail.com

Fig F.3: Result Of Simple View

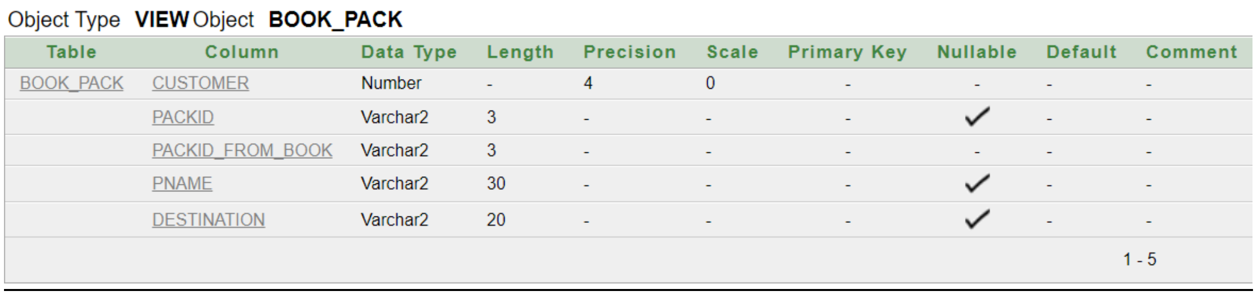
Complex View:

Create a complex view with euqi-join between book and package showing the customer id, package id, package name, destination with the results sorted by customer ID



```
create view book_pack (customer,packid,packid_from_book,pname,destination) as
select b.cus_id,b.p_id,p.p_id,p.p_name,p.destination
from books b,package p where b.p_id = p.p_id order by b.cus_id

describe book_pack
select *from book_pack
```

Fig F.4: Command Of Simple View

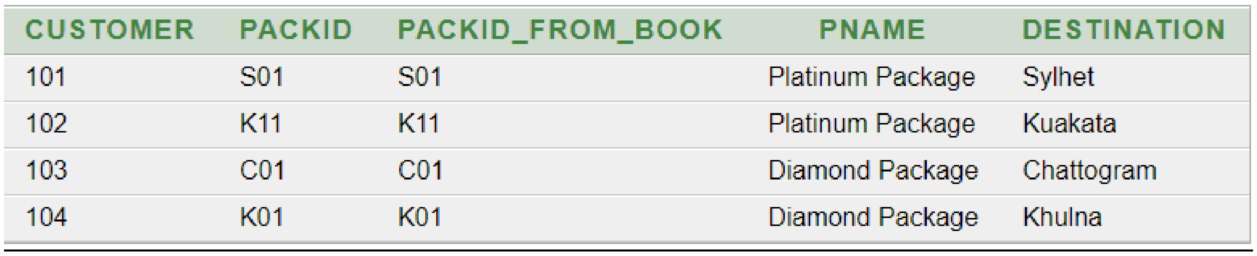


Object Type **VIEW** Object **BOOK_PACK**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BOOK_PACK	CUSTOMER	Number	-	4	0	-	-	-	-
	PACKID	Varchar2	3	-	-	-	✓	-	-
	PACKID_FROM_BOOK	Varchar2	3	-	-	-	-	-	-
	PNAME	Varchar2	30	-	-	-	✓	-	-
	DESTINATION	Varchar2	20	-	-	-	✓	-	-

1 - 5

Fig F.5: Description Of Simple View



CUSTOMER	PACKID	PACKID_FROM_BOOK	PNAME	DESTINATION
101	S01	S01	Platinum Package	Sylhet
102	K11	K11	Platinum Package	Kuakata
103	C01	C01	Diamond Package	Chattogram
104	K01	K01	Diamond Package	Khulna

Fig F.6: Result Of Simple View

9.Database Connection:

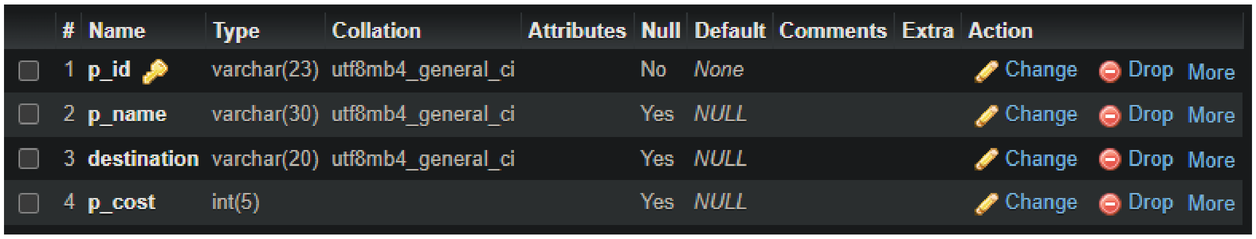
Connection No 1:

This report details the implementation of Java Database Connectivity (JDBC) in the context of the Travel and Tourism Management System. The objective is to connect to the MySQL database and retrieve information from the "Package" table. The code is executed within the IntelliJ IDEA integrated development environment.

The Java program utilizes JDBC to establish a connection to the MySQL database named "Travel and Tourism Management System." It specifically interacts with the "Package" table, extracting data for display. The code follows the standard JDBC workflow by loading the MySQL JDBC driver, creating a connection, and executing a SELECT query on the "Package" table. The retrieved information, including package details such as name, description, destination, and price, is then printed to the console.

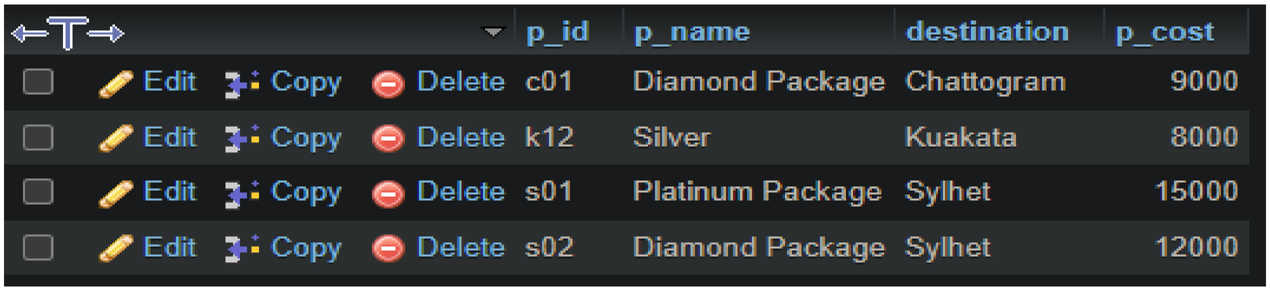
The development environment for this project is IntelliJ IDEA. The code is structured with a main class named `Conn`, which encapsulates the database connectivity logic. While the code successfully achieves its goal of displaying package information, it employs deprecated practices, such as not using try-with-resources for resource management. Additionally, the usage of an empty password for the database connection raises security concerns. It is recommended to adopt more modern and secure coding practices, including explicit resource closure and avoiding deprecated methods.

In summary, the Java program successfully connects to the "Travel and Tourism Management System" database, retrieves data from the "Package" table, and displays the details of various travel packages.



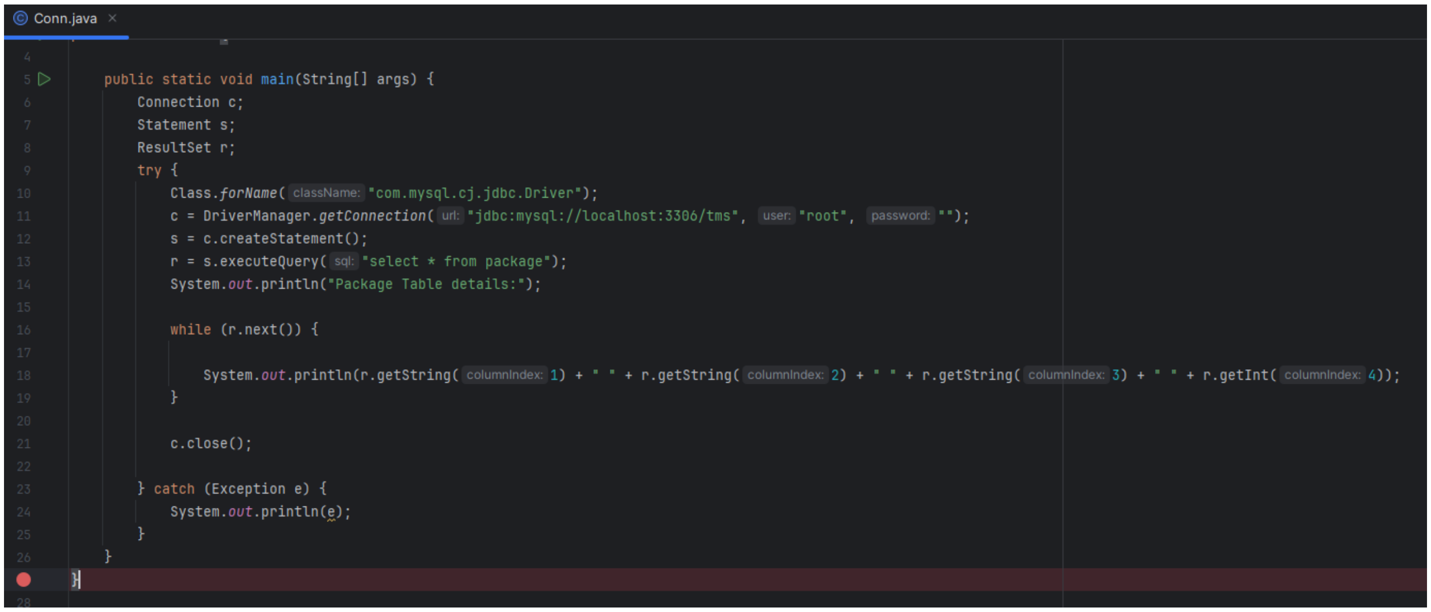
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	p_id	varchar(23)	utf8mb4_general_ci		No	None			Change Drop More
☐	2	p_name	varchar(30)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	3	destination	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
☐	4	p_cost	int(5)			Yes	NULL			Change Drop More

Figure 9.1.1: Description of Package Table from MySQL



				p_id	p_name	destination	p_cost
☐	Edit	Copy	Delete	c01	Diamond Package	Chattogram	9000
☐	Edit	Copy	Delete	k12	Silver	Kuakata	8000
☐	Edit	Copy	Delete	s01	Platinum Package	Sylhet	15000
☐	Edit	Copy	Delete	s02	Diamond Package	Sylhet	12000

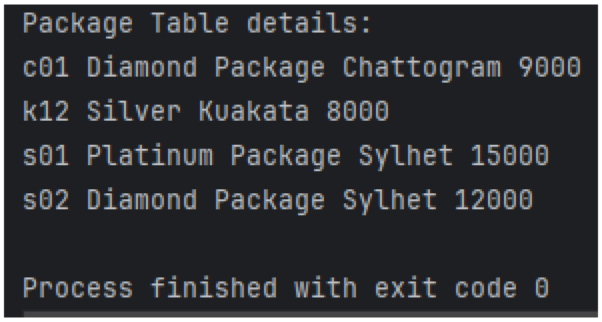
Figure 9.1.2: Package Table from MySQL



Conn.java

```java
    public static void main(String[] args) {
        Connection c;
        Statement s;
        ResultSet r;
        try {
            Class.forName( className: "com.mysql.cj.jdbc.Driver");
            c = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/tms",  user: "root",  password: "");
            s = c.createStatement();
            r = s.executeQuery( sql: "select * from package");
            System.out.println("Package Table details:");

            while (r.next()) {

                System.out.println(r.getString( columnIndex: 1) + " " + r.getString( columnIndex: 2) + " " + r.getString( columnIndex: 3) + " " + r.getInt( columnIndex: 4));
            }

            c.close();

        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Figure 9.1.3: Code for the connection between Java Program and Package Table



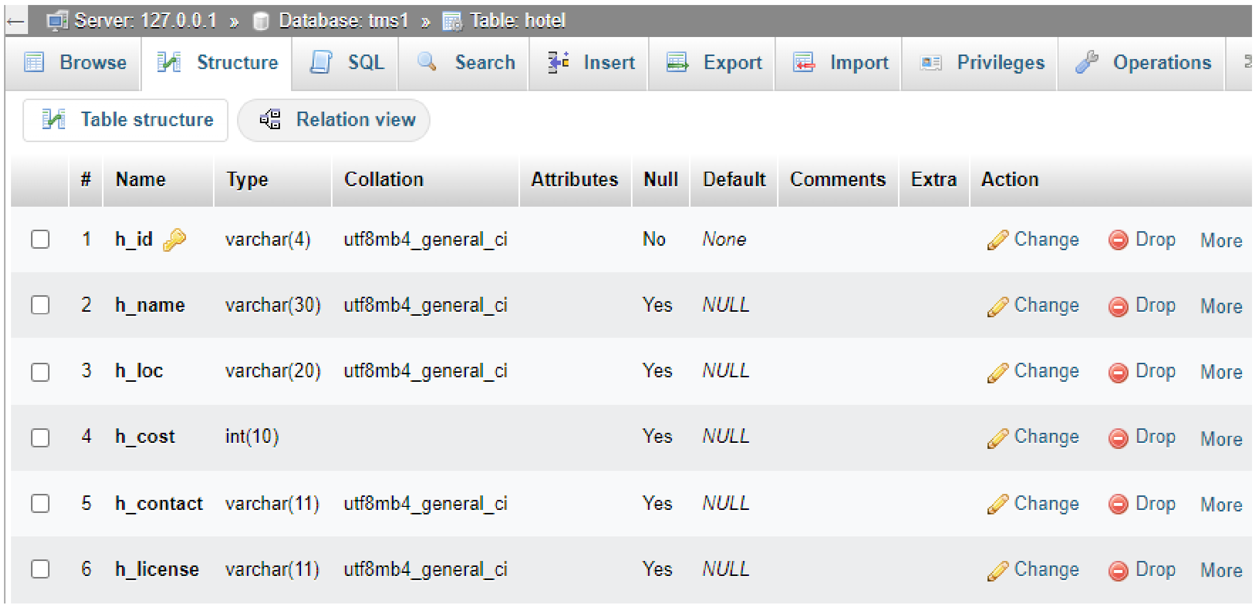
```
Package Table details:
c01 Diamond Package Chattogram 9000
k12 Silver Kuakata 8000
s01 Platinum Package Sylhet 15000
s02 Diamond Package Sylhet 12000

Process finished with exit code 0
```

Figure 9.1.4: Output of the Java Code

Connection No 2:

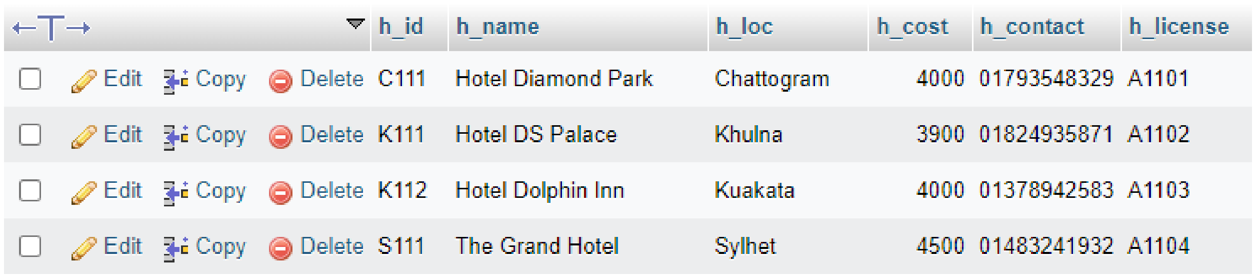
The provided Java program serves as a part of a **'Travel & Tourism Management System'**, specifically focusing on retrieving and displaying hotel details from a MySQL database. I use **'XAMPP Control Panel'** for retrieving data from MySQL database and **'IntelliJ IDEA'** to write the java code. The program establishes a database connection, executes a query to fetch hotel information, and then formats and prints the results in a tabular format.

Firstly, I use an import statement that brings in all the classes and interfaces from the java.sql package into the current Java source file. Then I write the basic syntax of a java program. Between a 'try-catch' block the **Connection** object (**con**) is created using the **DriverManager.getConnection** method, establishing a connection to the MySQL database named "**tms1**." The connection is made to the local server at port **3306**, with the username "**root**" and an **empty password**. Then A **Statement** object (**st**) is created from the connection to execute SQL queries. The **executeQuery** method is used to execute the SQL query (select * from hotel) to retrieve all records from the "**hotel**" table. The retrieved data is stored in a **ResultSet** object (**rs**), and a formatted table header is printed to the console. A **while loop** iterates through the **ResultSet** to fetch each row of hotel details. The **printf** method is utilized to format and print the hotel details in a tabular format, including columns for HID, Name, Location, Price, Contact, and License No. The program ensures proper resource management by closing the **ResultSet**, **Statement**, and **Connection** objects within a **try-catch** block to handle any potential exceptions. The **try-catch** block is used to catch and handle any exceptions that may occur during the execution of the code, ensuring graceful error handling.



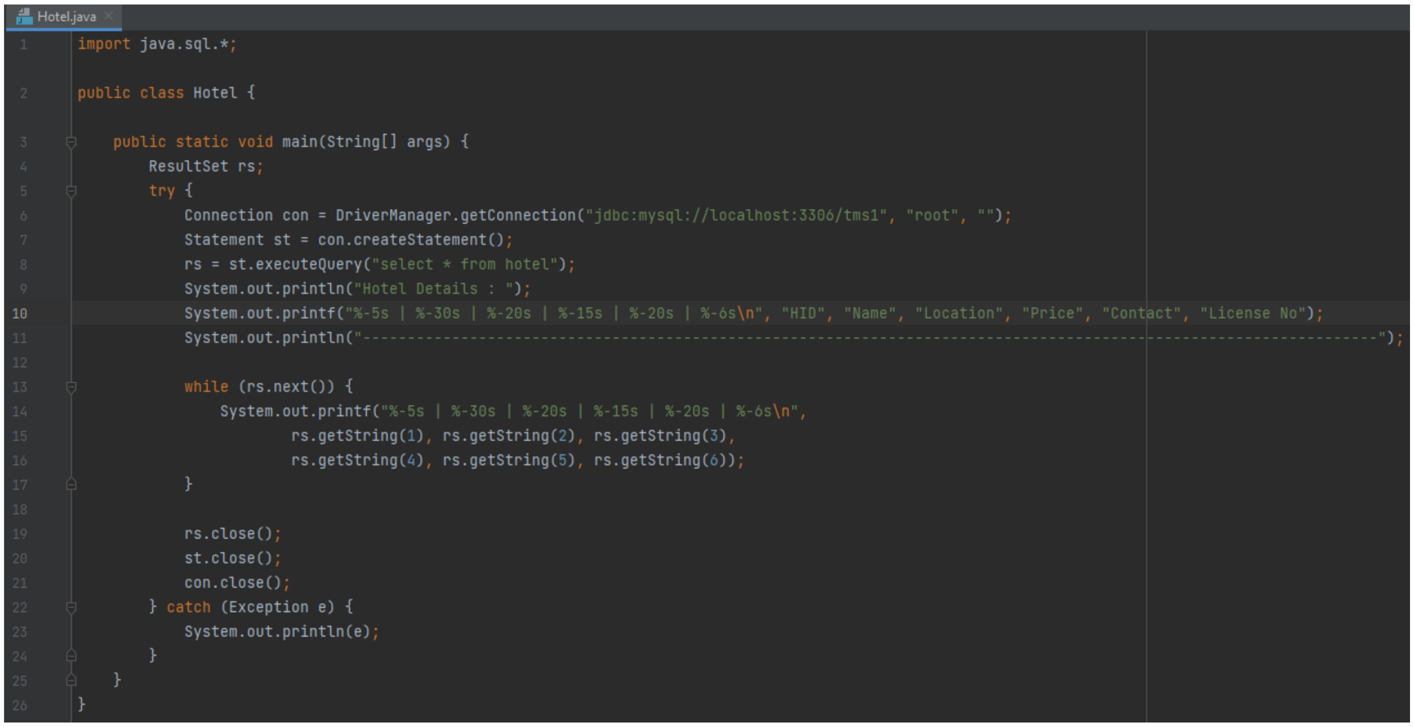
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	h_id	varchar(4)	utf8mb4_general_ci		No	None			Change Drop More
2	h_name	varchar(30)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	h_loc	varchar(20)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	h_cost	int(10)			Yes	NULL			Change Drop More
5	h_contact	varchar(11)	utf8mb4_general_ci		Yes	NULL			Change Drop More
6	h_license	varchar(11)	utf8mb4_general_ci		Yes	NULL			Change Drop More

Figure 9.2.1: Description of Hotel Table from MySQL



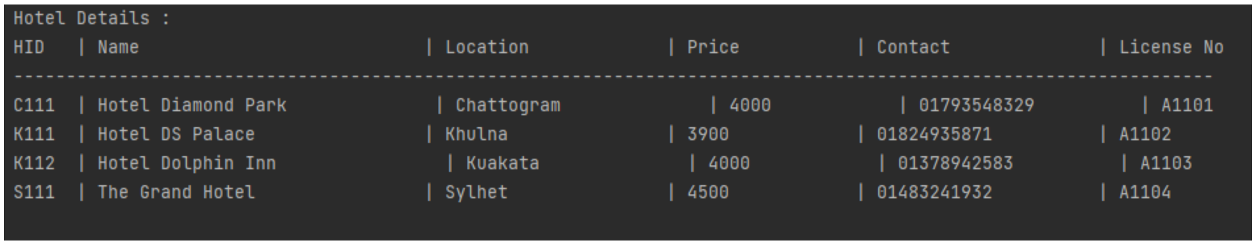
←T→				h_id	h_name	h_loc	h_cost	h_contact	h_license
☐	Edit	Copy	Delete	C111	Hotel Diamond Park	Chattogram	4000	01793548329	A1101
☐	Edit	Copy	Delete	K111	Hotel DS Palace	Khulna	3900	01824935871	A1102
☐	Edit	Copy	Delete	K112	Hotel Dolphin Inn	Kuakata	4000	01378942583	A1103
☐	Edit	Copy	Delete	S111	The Grand Hotel	Sylhet	4500	01483241932	A1104

Figure 9.2.2: Hotel Table from MySQL



```java
import java.sql.*;

public class Hotel {

    public static void main(String[] args) {
        ResultSet rs;
        try {
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/tms1", "root", "");
            Statement st = con.createStatement();
            rs = st.executeQuery("select * from hotel");
            System.out.println("Hotel Details : ");
            System.out.printf("%-5s | %-30s | %-20s | %-15s | %-20s | %-6s\n", "HID", "Name", "Location", "Price", "Contact", "License No");
            System.out.println("-------------------------------------------------------------------------------------------------------------------");

            while (rs.next()) {
                System.out.printf("%-5s | %-30s | %-20s | %-15s | %-20s | %-6s\n",
                        rs.getString(1), rs.getString(2), rs.getString(3),
                        rs.getString(4), rs.getString(5), rs.getString(6));
            }

            rs.close();
            st.close();
            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Figure 9.2.3: Code for the connection between Java Program and Hotel Table



```
Hotel Details :
HID   | Name                           | Location             | Price           | Contact              | License No
------------------------------------------------------------------------------------------------------------------
C111  | Hotel Diamond Park               | Chattogram             | 4000            | 01793548329          | A1101
K111  | Hotel DS Palace                | Khulna               | 3900            | 01824935871          | A1102
K112  | Hotel Dolphin Inn                 | Kuakata             | 4000            | 01378942583          | A1103
S111  | The Grand Hotel                | Sylhet               | 4500            | 01483241932          | A1104
```

Figure 9.2.4: Output of the Java Code

10.Conclusion:

“TRIPIFY – A TRAVEL & TOURISAM MANAGEMENT SYSTEM” using oracle 10g has proven to be an efficient solution that is designed with the primary objective of enhancing the overall travel experience for all users. By efficiently managing bookings, providing real-time information, and offering personalized recommendations, the system aims to streamline the travel process, ensuring convenience and satisfaction for every traveler. Through its user-friendly interface and comprehensive features, the database serves as a valuable tool in promoting seamless journeys and creating a positive impact on the travel and tourism industry.