HOTEL CHAIN MANAGEMENT SYSTEM

MYSQL QUERIES ON NORMALIZATION PROJECT

GROUP 16

Nimra Ashraf & Noor Fatima

110829 & 110830

MORNING

Submitted to:

Mam Sehrish Khan

11/05/2025

TABLE OF CONTENTS

DESCRIPTION

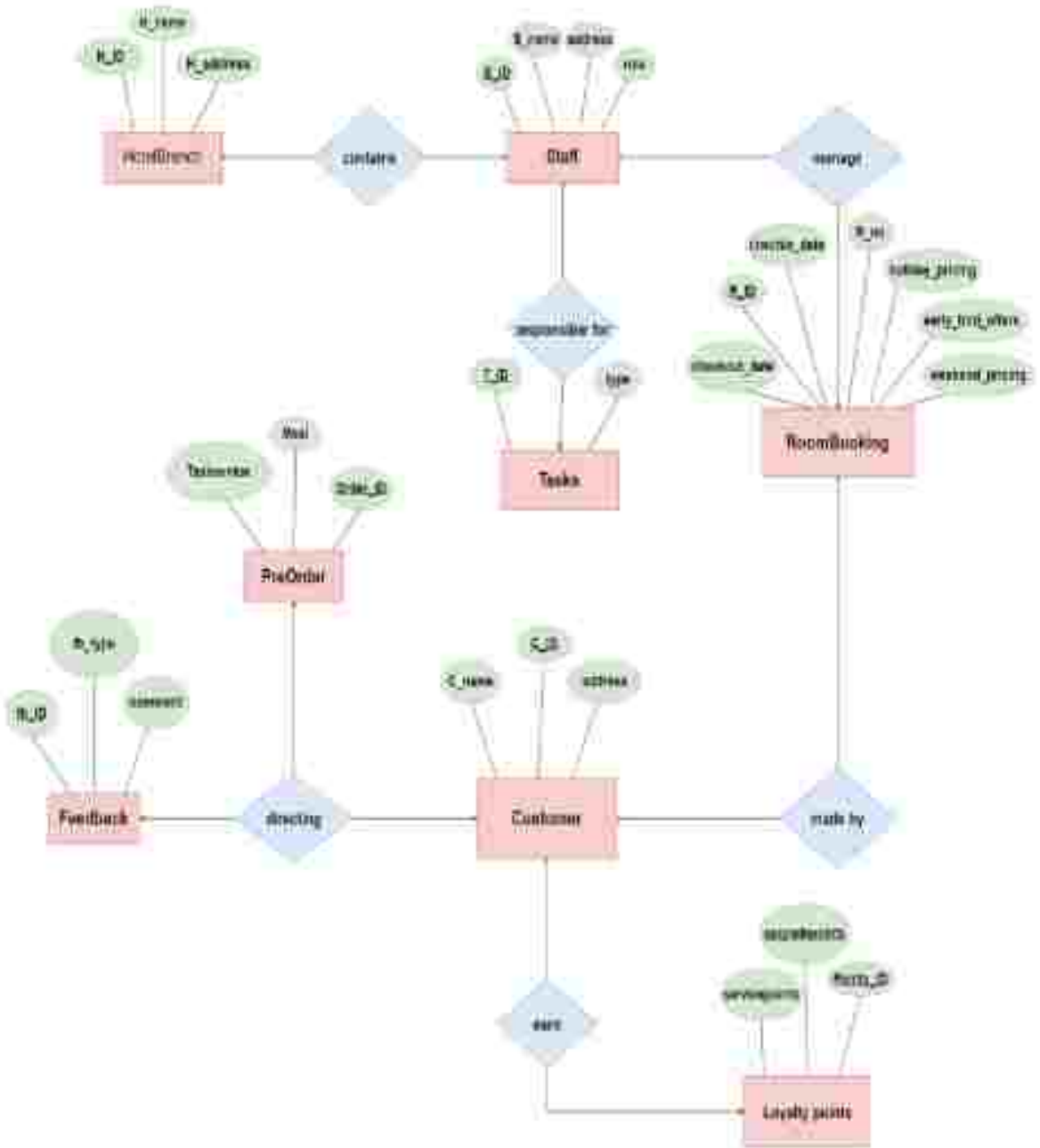
SQL stands for structure query language. It is a widely used, open source relational database management system.

A hotel chain management system provides a centralized platform for managing multiple hotel properties within a chain. This database would store information about different hotels, including room details, staff managing, reservations, guest records pricing information etc.

Pros:

- Scalability
- Reliability
- Cost effectiveness
- Flexibility

ENTITY RELATIONSHIP DIAGRAM



⇒ RELATIONAL SCHEMA

1. **hotel_branch** (h_id, h_name, h_address, staff_id)
2. **staff** (s_id, s_name, s_address, s_role)
3. **room_booking** (R_id, R_no, check_in_date, check_out_date, weekend_pricing, holiday_pricing, early_bird_offers, s_id)
4. **customer** (c_id, c_name, c_address, r_id)
5. **loyalty_points** (points_id, upgrade_points, service_points, cus_id)
6. **feedback** (f_id, f_type, comment, customer_id)
7. **pre_order** (order_id, meal, taxi_service, cust_id)
8. **task** (t_id, type, s_id)

SQL QUERIES

Show DATABASES

Show all databases and tables placed in a database

MySQL use the following command:

⇒ **show databases;**

⇒ **show tables;**

CREATE DATABASE & TABLES

Create a new database or table.

⇒ **create database;**

⇒ **create table table_name (attribute datatype (size),**

...);

USING DATABASE

Use a database already saved in MySQL.

⇒ **use database database_name;**

DESCRIBE TABLES

To see the constraints we have assigned to tables,

DESCRIBE keyword is used.

⇒ **DESCRIBE table_name;**

SELECT

Used to retrieve rows selected from one or more tables.

⇒ **Select * from table_name;**

⇒ **Select attribute1_name from table_name where attribute2_name='---' ;**

INSERT INTO

Insert values in tables.

⇒ **INSERT INTO table_name values ('attribute' datatype (value), ...);**

ALTER TABLES

Used for many purposes such as:

1. To rename a column of a table

⇒ **ALTER table table_name RENAME COLUMN column_name from existing_name to new_name;**

2. To add a new column in a table

⇒ **ALTER table table_name ADD column datatype (size);**

3. To make an attribute foreign key

⇒ **ALTER table table_name ADD column_name FOREIGN KEY REFERENCES referencetable_name (P.K);**

TABLE UPDATION

To reset the values of attributes in a table.

⇒ **Update table_name set attribute='value' WHERE attribute_PK='target-value';**

TABLE CONTRAINTS

There are two constraints for tables, Primary key and Foreign key.

Primary Key:

The attribute of a table on which all the other attributes of that depend.

Foreign Key:

When Primary key of a table is used in another table, it becomes Foreign key.

⇒ **ALTER table table_name ADD FOREIGN KEY (key_name) REFERENCES reference_table (P.K);**

Arithmetic Operations

Arithmetic operations include operators such as +, -, *, / etc. to perform addition, subtraction, multiplication and division of two attributes.

⇒ **select attribute operator value from table;**

Logical / Relational Operations

Logical operations include operators such as >, <, <=, >=, !=, == etc. to relate any two attributes of a table.

⇒ **select attribute1 from table_name where attribute2 relational operator value;**

Aggregation Functions

Aggregate functions include avg, max, min etc. operations.

⇒ **select aggregate_function (attribute) from table;**

ORDER BY

ORDER BY keyword is used to sort the values of

tables in ascending or descending order. By default ascending order is set. Otherwise for sorting the records in descending order DESC order is used. For ascending ASC and descending DESC is used.

Syntaxes of above two orders are:

⇒ **select * from table ORDER BY attribute DESC;**

⇒ **select * from table ORDER BY attribute ASC;**

GROUP BY

GROUP BY clause is very important used to group rows from a table based on the values of one or more column. It is used with aggregate functions like **AVG, MAX, MIN, SUM and COUNT** to perform calculations on grouped data. Also we can perform operations on group within the group.

Syntax:

⇒ **select aggreagate_function (attribute) from table GROUP BY attributes;**

AS

The AS keyword in MySQL is used to assign an alias to a table or column, making it easier to reference or improving readability. It allows for temporary renaming with a query, which can simplify complex queries and result sets.

Where

This clause is used to filter records. It is used to extract only those records that fulfill a specified condition.

DISTINCT

The DISTINCT keyword in MySQL is used to

remove duplicate records from the results of a SELECT query. It ensures that the query returns only unique values in the specified columns.

Syntax:

⇒ select DISTINCT attribute from table;

BETWEEN

The BETWEEN clause is used to show the values/ contents of the table between a given limit. It filter the result set within a specified range.

Syntax:

⇒ **select column_name from table where column_name BETWEEN value1 AND value2;**

Count & Count(*)

Used to count the number of rows in a table. This function counts all rows regardless of whether they contain NULL values.

Syntax:

⇒ **select count (*) attribute_name from table;**

⇒ **select count (attribute_name) from table;**

HAVING

The HAVING clause is similar to the WHERE clause but is specifically applied after grouping and aggregation, allowing you to filter on the results of

aggregate functions like COUNT, SUM, AVG, and others.

Syntax:

⇒ **select aggreagate_function (attribute) from table GROUP BY attributes HAVING count (attribute) >1;**

AND & OR

The AND & OR operators are used to filter records based on more than one conditions:

- The AND operator displays a record if all the conditions separated by AND are true.
- The AND operator displays a record if any of the conditions separated by AND are true.

IN

The IN operator allows you to specify multiple values in a where clause. It is a shorthand for multiple OR conditions.

⇒ **SELECT attribute_name(s) FROM table_name WHERE attribute_name IN (value1, value2, ...);**

LIKE

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

Syntax:

⇒ **SELECT column1. . .FROM table_name**
WHERE columnN LIKE pattern;

1. The percent sign (%) represents zero, one, or multiple characters
2. The underscore sign (_) represents one, single character

IS NULL & IS NOT NULL

These keywords are used for checking that the values of attributes checked are **NULL** or not.

Syntax:

⇒ **select attribute_name from table where attribute IS NULL;**

⇒ **select attribute_name from table where attribute IS NOT NULL;**

JOINS

Joins allows to retrieve related data from multiple tables in a single query, avoiding the need for different separate queries. There are multiple types of joins such as inner, right, left etc.

Syntax:

⇒ **Select column_list from table1 JOIN table2 ON table1. column = table1;**

INNER JOIN:

The joins in which both the tables have matching values in them are called inner join.

Left Join:

This join return all rows from the left table and matching rows in right table.

Cross join:

A cross join is type of join that return cartesian product of rows from the tables in the join .

Equi join:

It is join operation in sql that combines two table based on a matching column between them.

Right Join:

This join return all rows from right table and matching rows from left table.

VIEWS

A MySQL view is a predefined select query that operates on existing data without duplicating it. A view acts as a virtual table.

Syntax:

⇒ **create or replace VIEW view_name AS select column1, column2 from table_name where condition;**

DELETE

DELETE statement is used to delete rows in a table.

It deletes a specific row using where clause.

Syntax:

⇒ **delete from table where column_name=**

'value';

DROP

DROP statement is used to delete the whole table along with table structure, attribute and indexes.

Syntax:

⇒ **drop table table_name;**

TRUNCATE

The truncate statement is used to delete all data in the table not the whole table.

Syntax:

⇒ **truncate table_name;**

Sub Query

Subqueries are also known as inner queries or nested queries. It is embedded inside another query and acts as input or output for that query.

Syntax:

⇒ **Select column1, column2... from table where column operator (select column from another_table where condition); column1, column2, ...: The columns you want to retrieve.**

GRANT

Grant is a statement used to assign privileges to user accounts, allowing them to perform specific actions on database projects.

PRIVILEGES

Privileges are the rights or permissions assigned to users that determine what actions they can perform on the database.

NORMALIZATION TABLES

CREATE DATABASE HOTEL_CHAIN

CREATE Hotel_Branch Table

INSERTING VALUES IN HOTEL_BRANCH

3 NF OF HOTEL_BRANCH

CREATE STAFF TABLE

INSERTING VALUES IN STAFF

3RD NORMAL FORM OF STAFF

ADD s_id COLUMN IN HOTEL BRANCH TABLE

CHANGE COLUMN NAME (FROM s_id TO staff_id)

MAKE staff_id FOREIGN KEY

INSERTING VALUES IN FOREIGN KEY

SELECT FROM HOTEL_BRANCH

CREATE ROOM_BOOKING TABLE

MODIFY CHECK_IN & CHECK_OUT DATES

DESCRIBE ROOM_BOOKING

```
mysql> describe room_booking;
+-------------------+------+------+-----+---------+-------+
| Field             | Type | Null | Key | Default | Extra |
+-------------------+------+------+-----+---------+-------+
| room_id           | int  | NO   | PRI | NULL    |       |
| r_no              | int  | YES  |     | NULL    |       |
| check_in_date     | date | YES  |     | NULL    |       |
| check_out_date    | date | YES  |     | NULL    |       |
| weekend_pricing   | int  | YES  |     | NULL    |       |
| holiday_pricing   | int  | YES  |     | NULL    |       |
| early_bird_offers | int  | YES  |     | NULL    |       |
+-------------------+------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

MODIFY EARLY_BIRD_OFFERS & INSERTING VALUES

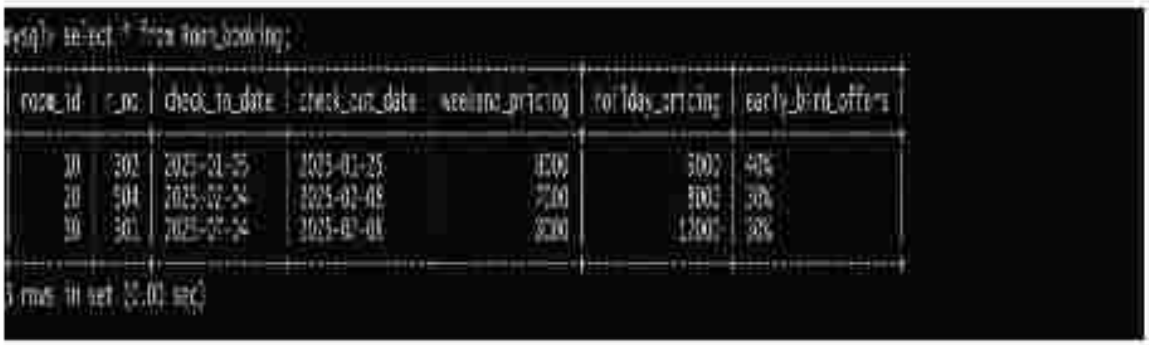
```
mysql> ALTER table Room_booking MODIFY COLUMN early_bird_offers varchar(20);
Query OK, 0 rows affected (0.18 sec)

mysql> INSERT into Room_booking values('10', '202', '2025-01-01', '2025-01-15', '3005', '9000', '40%');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT into Room_booking values('20', '304', '2025-02-4', '2025-02-8', '7010', '8500', '50%');
Query OK, 1 row affected (0.07 sec)

mysql> INSERT into Room_booking values('30', '301', '2025-02-5', '2025-02-8', '6010', '13002', '30%');
Query OK, 1 row affected (0.01 sec)
```

SELECT FROM ROOM_BOOKING



```
mysql> select * from Room_booking;
+---------+------+--------------+--------------+-----------------+-----------------+-------------------+
| room_id | r_no | check_in_date | check_out_date | weekend_pricing | holiday_pricing | early_bird_offers |
+---------+------+--------------+--------------+-----------------+-----------------+-------------------+
|      10 |  202 | 2025-01-15   | 2025-01-25   |            4200 |            5002 | 40%               |
|      20 |  304 | 2025-02-04   | 2025-02-08   |            7500 |            8002 | 50%               |
|      30 |  301 | 2025-02-04   | 2025-02-08   |            8200 |           13002 | 30%               |
+---------+------+--------------+--------------+-----------------+-----------------+-------------------+
3 rows in set (0.00 sec)
```

ADD STAFF_ID FOREIGN KEY IN ROOM_BOOKING

```
mysql> ALTER table room_booking ADD FOREIGN KEY (STAFF_ID) REFERENCES staffs(_id);
Query OK, 3 rows affected (0.12 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> DESCRIBE room_booking;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| room_id         | int         | NO   | PRI | NULL    |       |
| r_no            | int         | YES  |     | NULL    |       |
| check_in_date   | date        | YES  |     | NULL    |       |
| check_out_date  | date        | YES  |     | NULL    |       |
| weekend_pricing | int         | YES  |     | NULL    |       |
| holiday_pricing | int         | YES  |     | NULL    |       |
| early_bird_offers | varchar(20) | YES  |     | NULL    |       |
| STAFF_ID        | int         | YES  | MUL | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

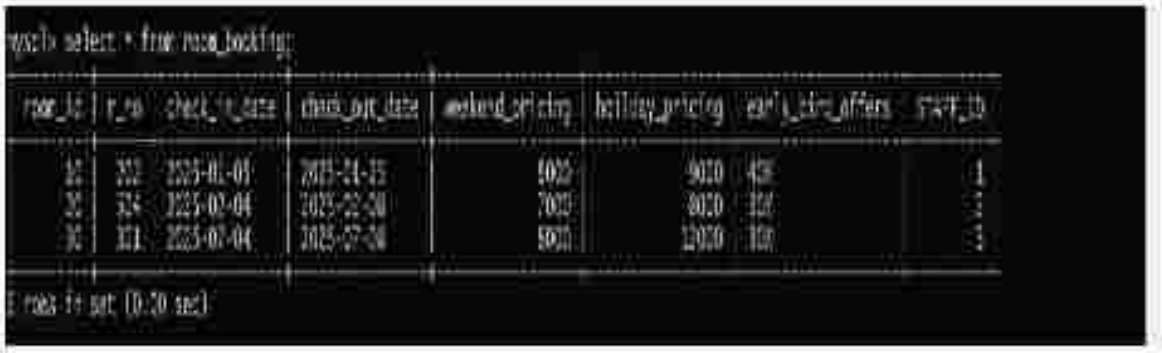
INSERT VALUES IN FOREIGN KEY

```
mysql> update room_booking set STAFF_ID='1' where room_id='10';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update room_booking set STAFF_ID='2' where room_id='20';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> update room_booking set STAFF_ID='3' where room_id='30';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

SELECT FROM ROOM_BOOKING



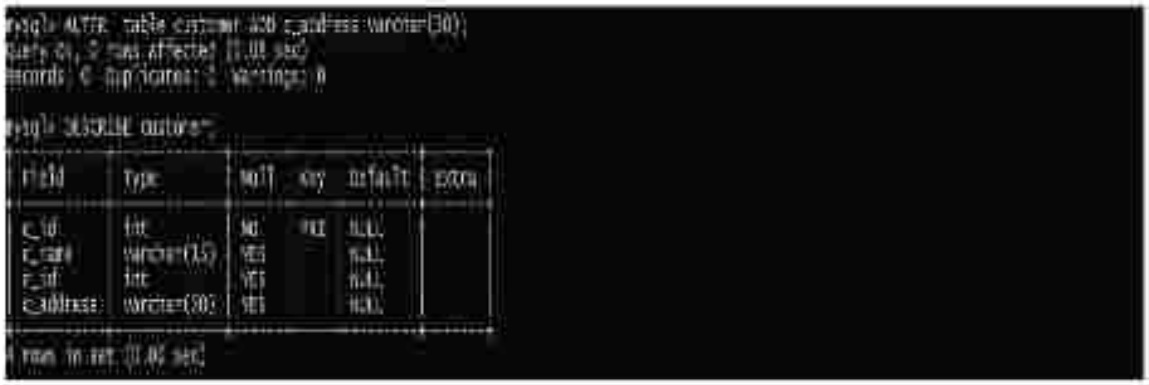
```
mysql> select * from room_booking;
+---------+------+---------------+----------------+-----------------+-----------------+-------------------+----------+
| room_id | r_no | check_in_date | check_out_date | weekend_pricing | holiday_pricing | early_bird_offers | STAFF_ID |
+---------+------+---------------+----------------+-----------------+-----------------+-------------------+----------+
|      10 |  222 | 2025-01-01    | 2025-24-25     |            5000 |            9000 | 45%               |        1 |
|      20 |  334 | 2025-02-04    | 2025-22-20     |            7000 |            8000 | 35%               |        2 |
|      30 |  321 | 2025-02-04    | 2025-27-28     |            6000 |           12000 | 10%               |        3 |
+---------+------+---------------+----------------+-----------------+-----------------+-------------------+----------+
3 rows in set (0.20 sec)
```

3 NF OF ROOM_BOOKING

MAKE ROOM_ID FOREIGN KEY IN STAFF TABLE

CREATE CUSTOMER TABLE

ADD MISING C_ADDRESS COLUMN IN CUSTOMER TABLE



INSERTING VALUES

MAKE r_id FOREIGN KEY IN CUSTOMER TABLE

CREATE LOYALTY POINTS TABLE

```
mysql> create table loyalty_points(point_id int(10) PRIMARY KEY, service_points INT(5), upgrade_points INT(5),cus_id INT(10));
Query OK, 0 rows affected, 4 warnings (0.14 sec)

mysql> DESCRIBE loyalty_points;
+----------------+------+------+-----+---------+-------+
| Field          | Type | Null | Key | Default | Extra |
+----------------+------+------+-----+---------+-------+
| point_id       | int  | NO   | PRI | NULL    |       |
| service_points | int  | YES  |     | NULL    |       |
| upgrade_points | int  | YES  |     | NULL    |       |
| cus_id         | int  | YES  |     | NULL    |       |
+----------------+------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

MAKE cus_id FOREIGN KEY

```
mysql> ALTER table loyalty_points ADD FOREIGN KEY (cus_id) REFERENCES customer (c_id);
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> describe loyalty_points;
+----------------+------+------+-----+---------+-------+
| Field          | Type | Null | Key | Default | Extra |
+----------------+------+------+-----+---------+-------+
| point_id       | int  | NO   | PRI | NULL    |       |
| service_points | int  | YES  |     | NULL    |       |
| upgrade_points | int  | YES  |     | NULL    |       |
| cus_id         | int  | YES  | MUL | NULL    |       |
+----------------+------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

INSERTING VALUES & DESCRIBE

```
mysql> INSERT INTO loyalty_points VALUES('154','300','50','120');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO loyalty_points VALUES('250','500','100','200'),('310','400','10','350');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from loyalty_points;
+----------+----------------+----------------+--------+
| point_id | service_points | upgrade_points | cus_id |
+----------+----------------+----------------+--------+
|      154 |            300 |             50 |    120 |
|      250 |            500 |            100 |    200 |
|      310 |            400 |             10 |    350 |
+----------+----------------+----------------+--------+
3 rows in set (0.00 sec)
```

CREATE FEEDBACK TABLE

MAKE customer_id FOREIGN KEY IN FEEDBACK TABLE

INSERTING VALUES

CREATE PRE_ORDER TABLE

Field	Type	Null	Key	Default	Extra
order_id	int	NO	PRI	NULL	
meal	varchar(40)	YES		NULL	
food_services	varchar(10)	YES		NULL	
cus_id	int	YES		NULL	

MADE cus_id FOREIGN KEY

Field	Type	Null	Key	Default	Extra
order_id	int	NO	PRI	NULL	
meal	varchar(40)	YES		NULL	
food_services	varchar(10)	YES		NULL	
cus_id	int	YES	MUL	NULL	

INSERTING VALUES IN ORDER

order_id	meal	food_services	cus_id
110	breakfast	yes	103
111	dinner	no	103
116	lunch	yes	103

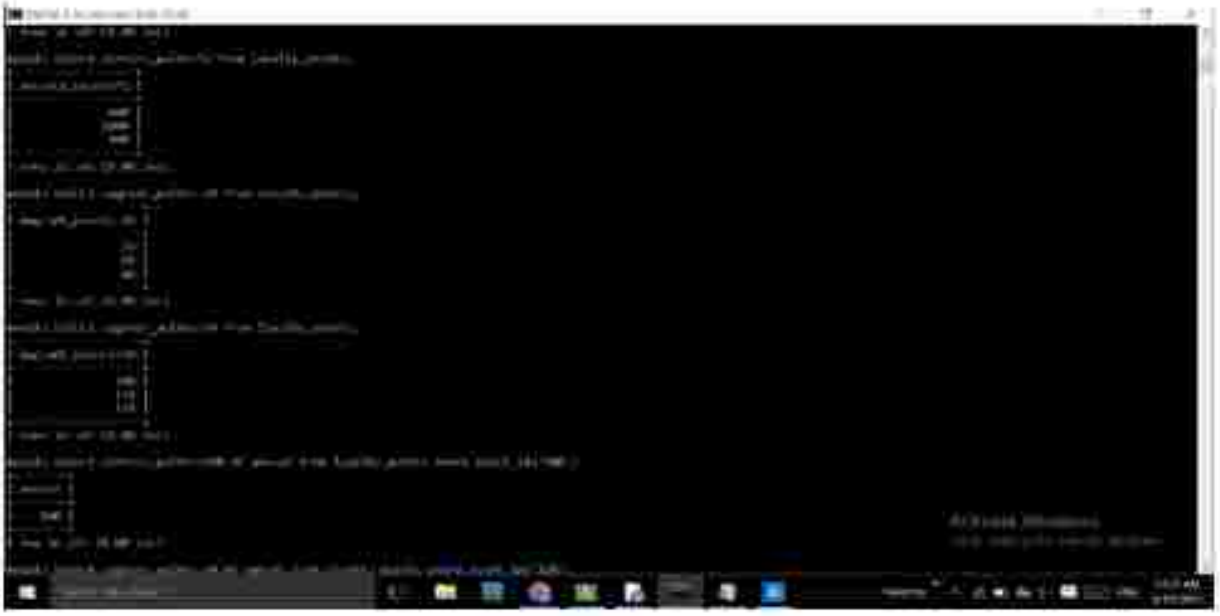
CREATE TASK TABLE

ADD st_id FOREIGN KEY AND DESCRIBE IT

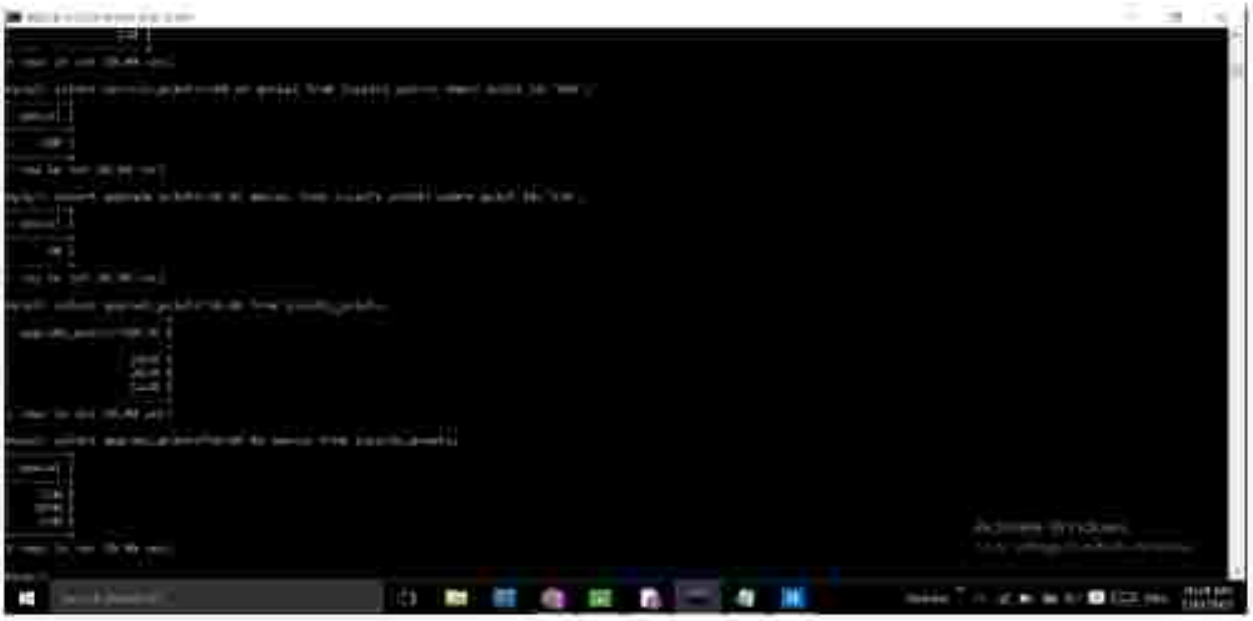
INSERTING VALUES IN TASK

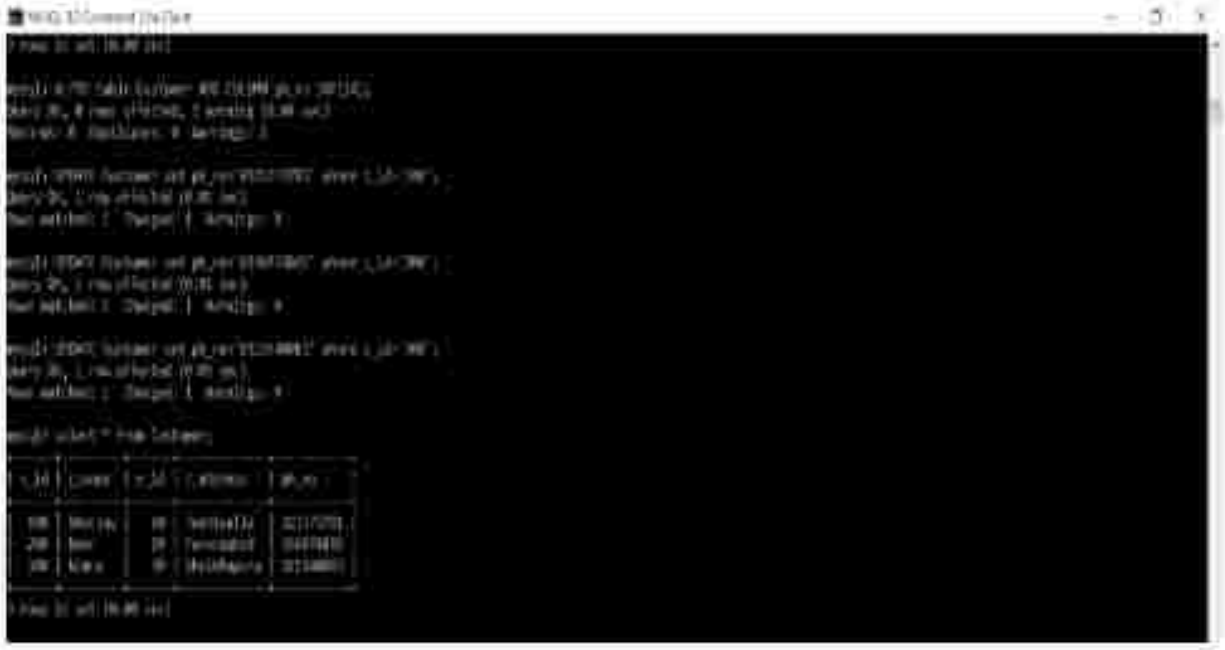
PRACTICAL IMPLEMENTATION OF MYSQL QUERIES

ARITHMETIC & LOGICAL OPERATIONS ON LOYALTY_POINTS TABLE

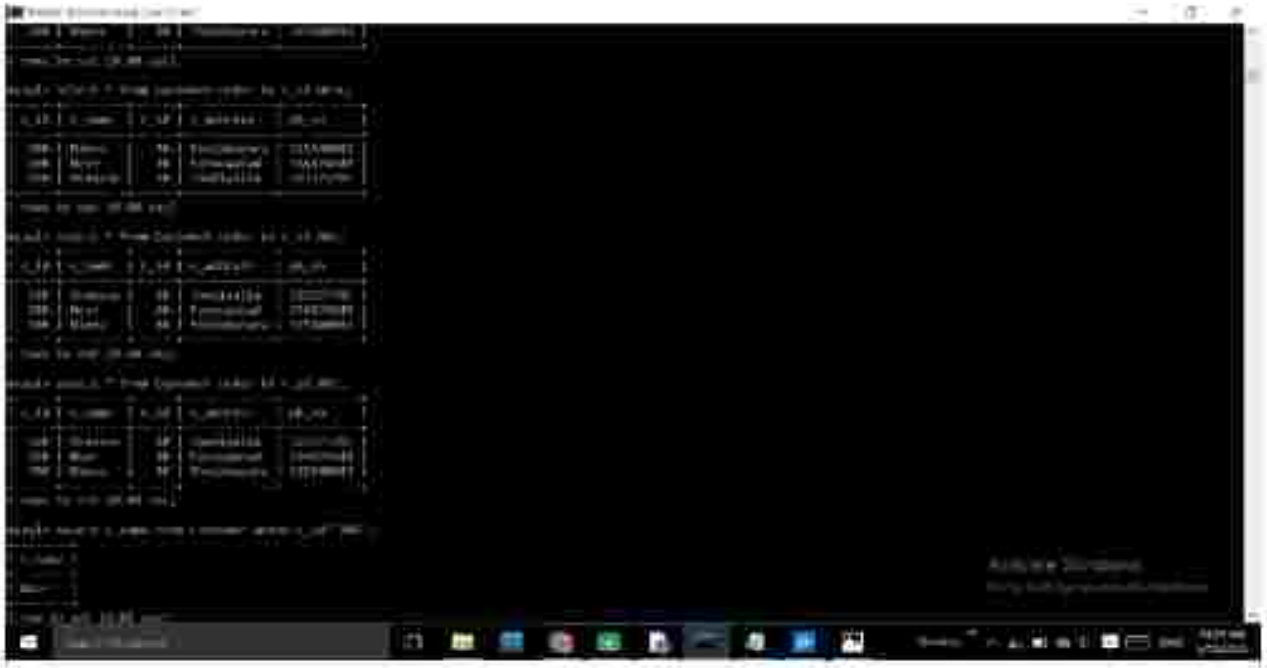


LOGICAL OPERATIONS ADDING ph_no COLUMN IN CUSTOMER

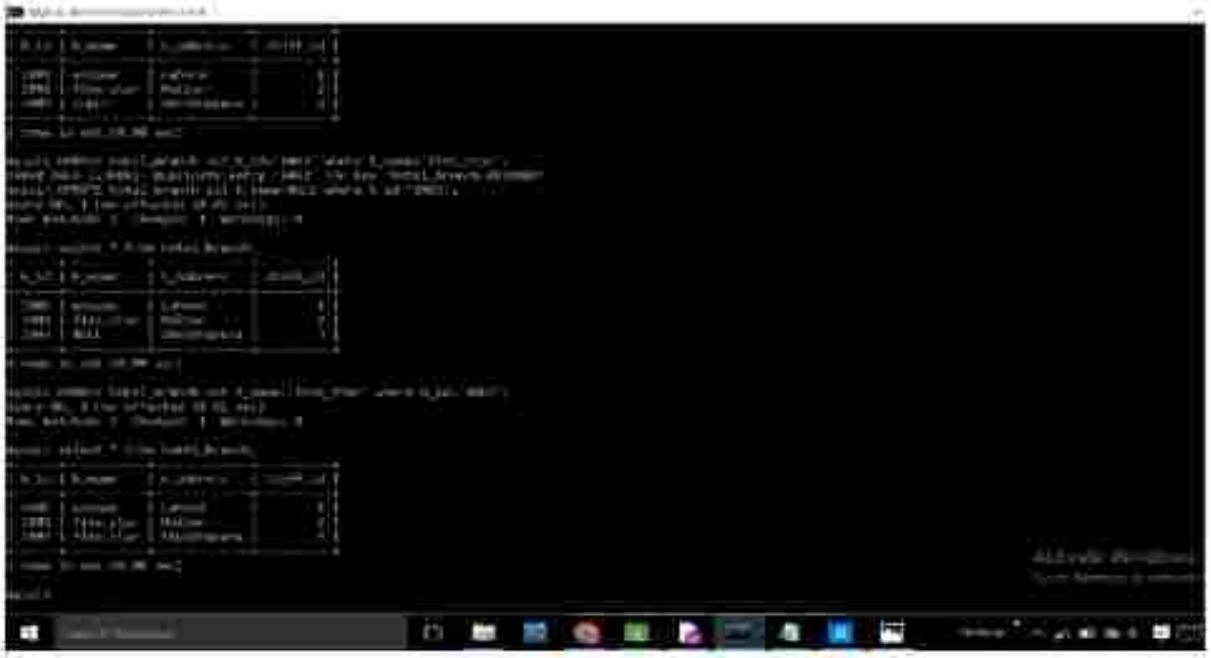




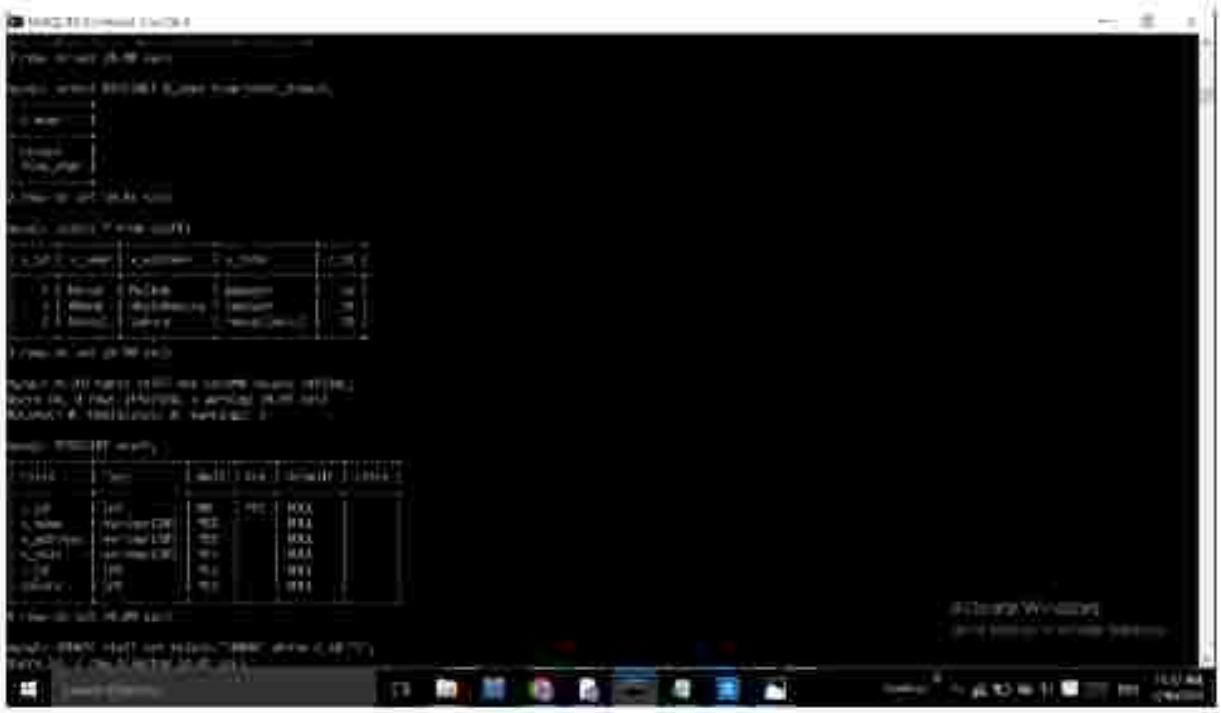
ORDER BY CLAUSE ON CUSTOMER



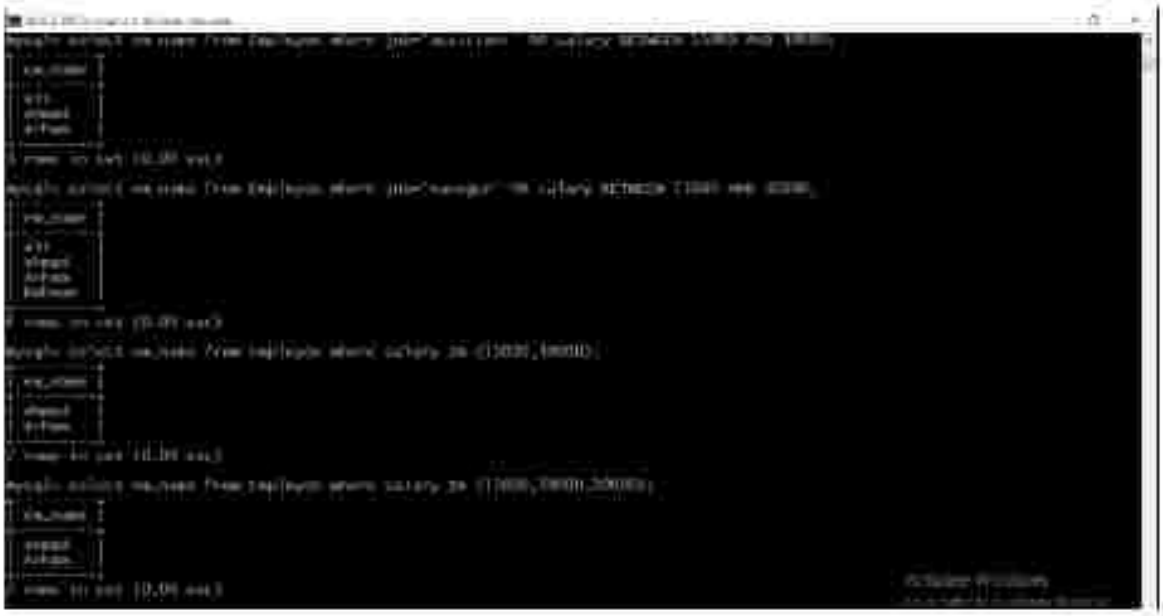
GIVING TWO h_ids SAME NAME

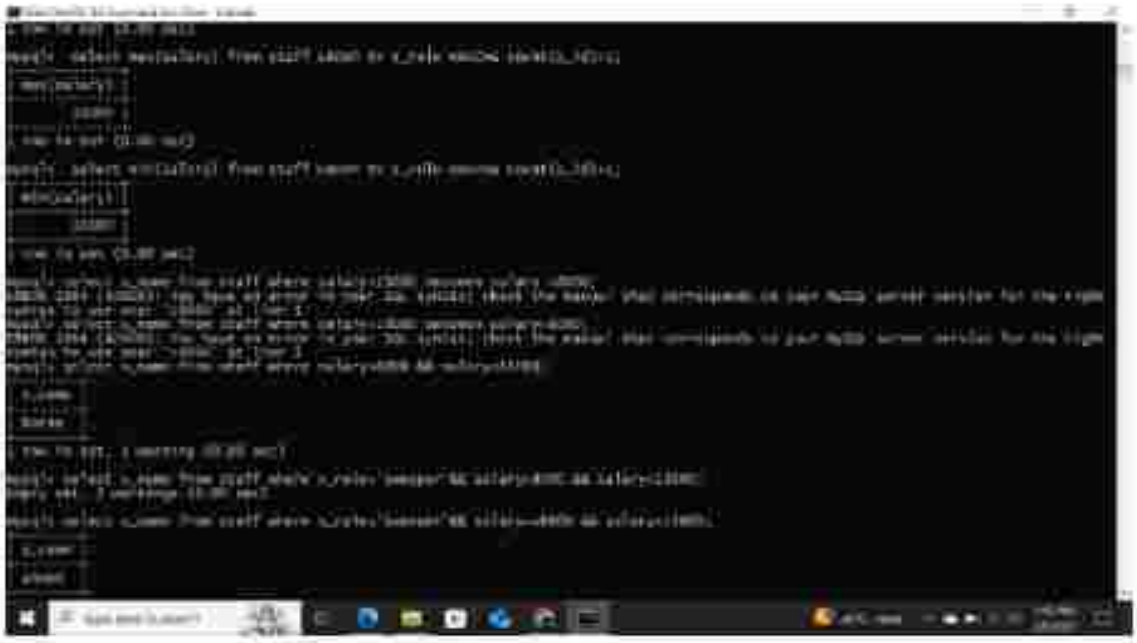


DISTINCT CLAUSE AND ADDING SALARY COLUMN IN STAFF TABLE

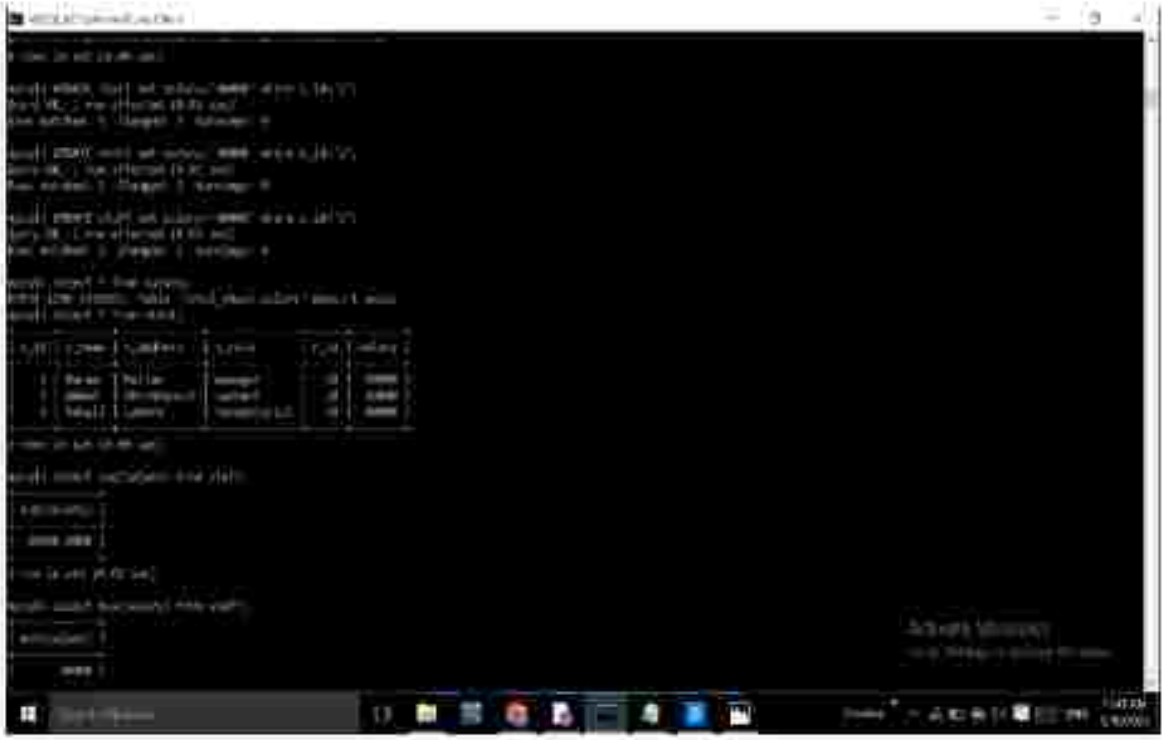


BETWEEN, HAVING AND IN CLAUSES



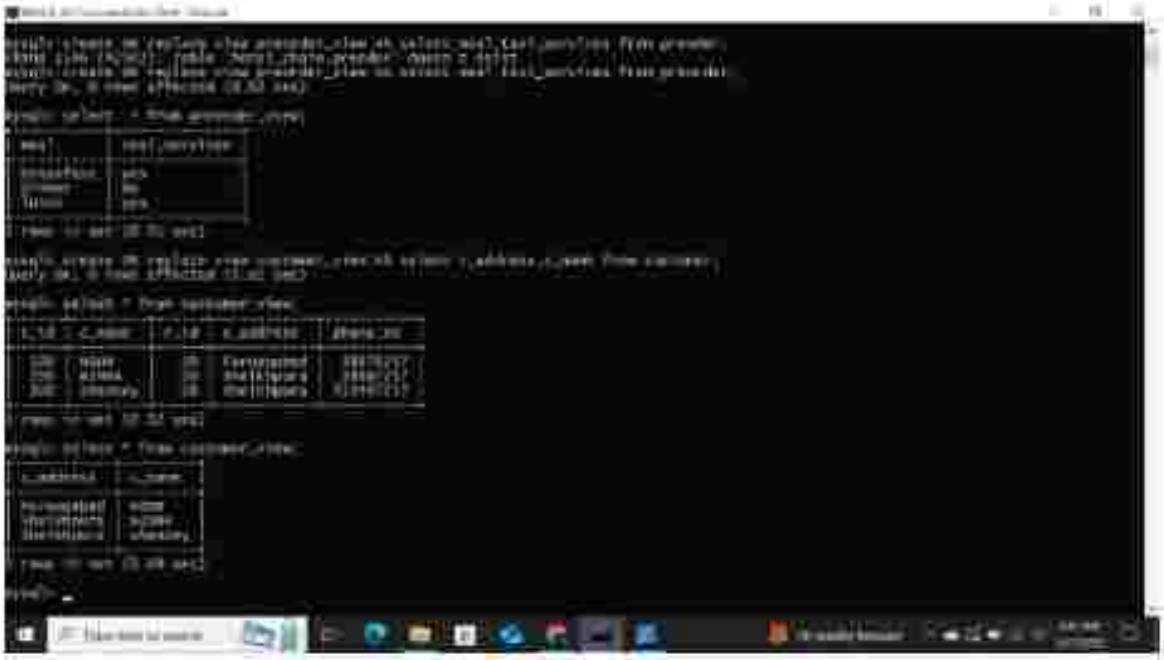


ARITHMETIC OPERATIONS ON STAFF



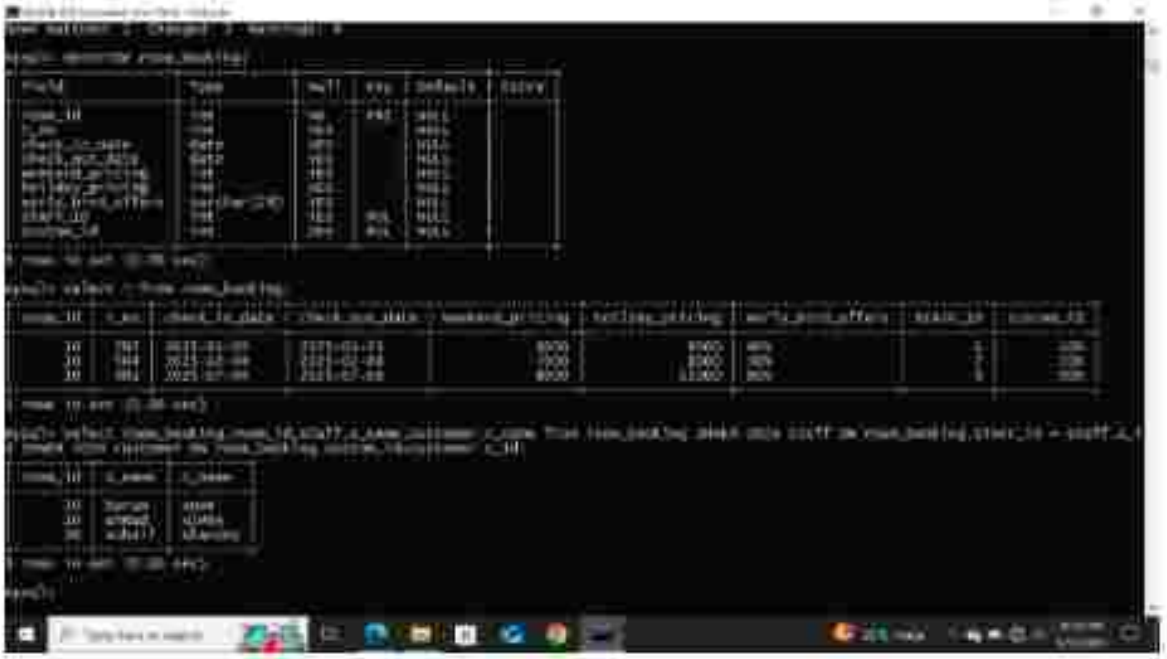
COUNT & COUNT(*)

VIEW

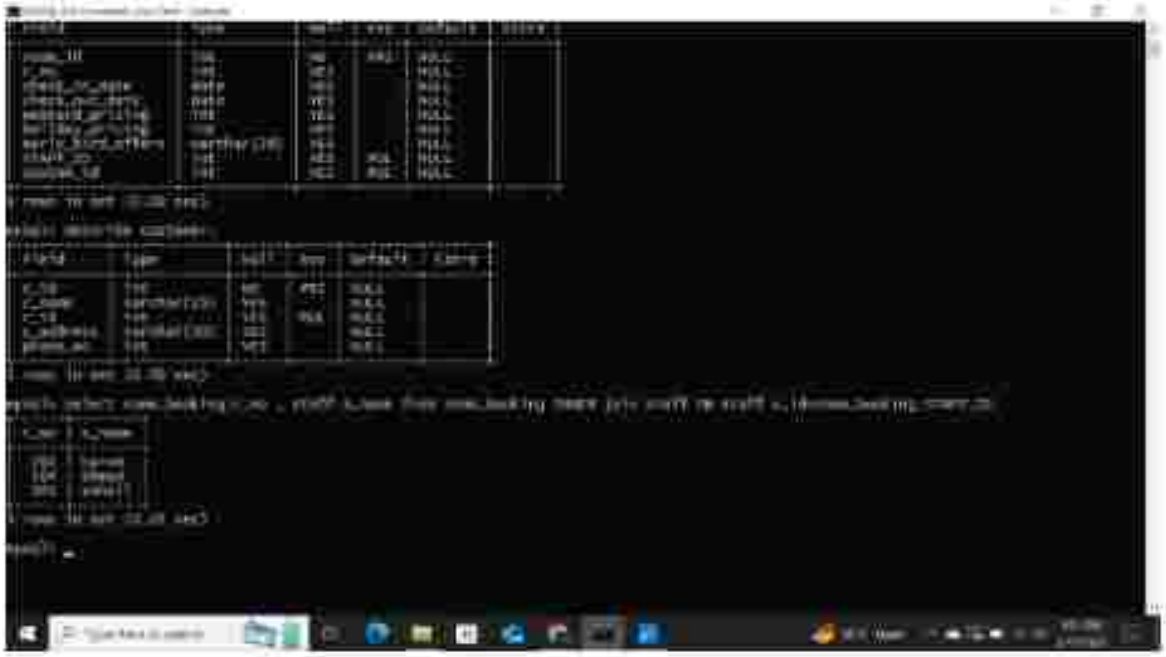


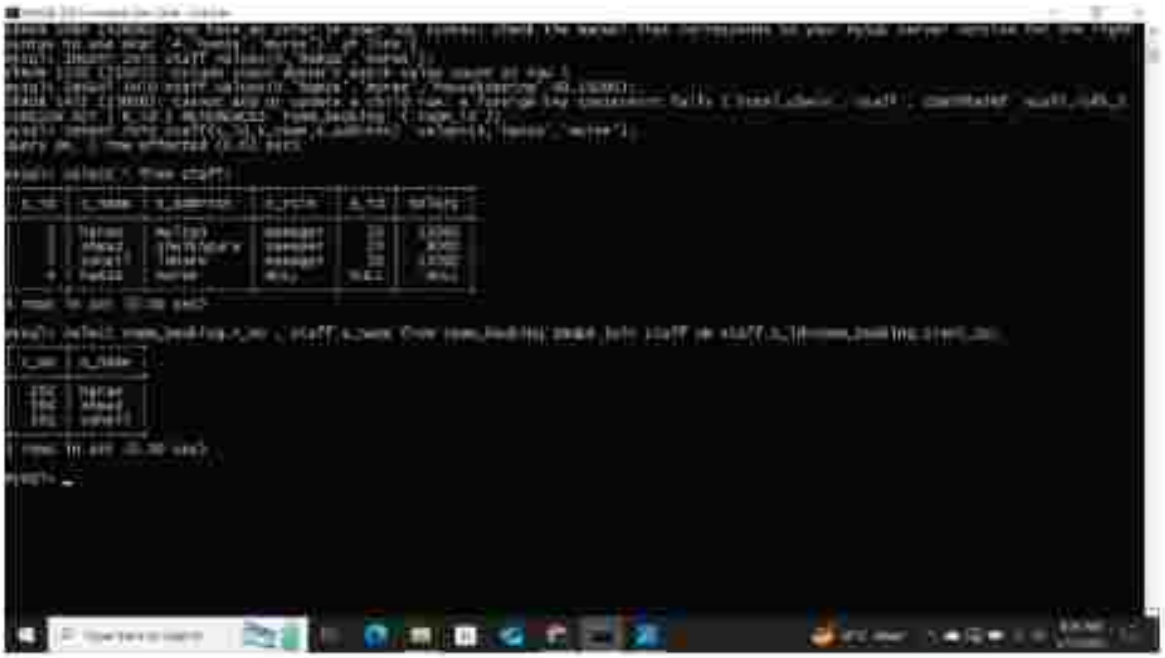
LIKE CLAUSE ON CUSTOMER

JOINS



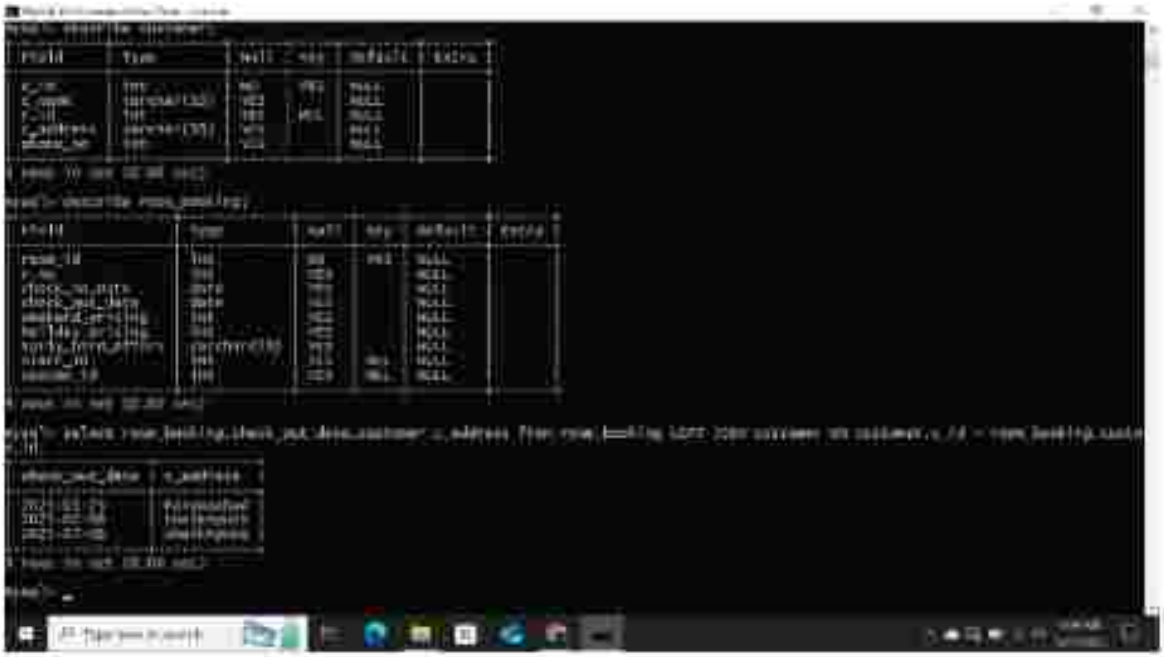
INNER JOIN





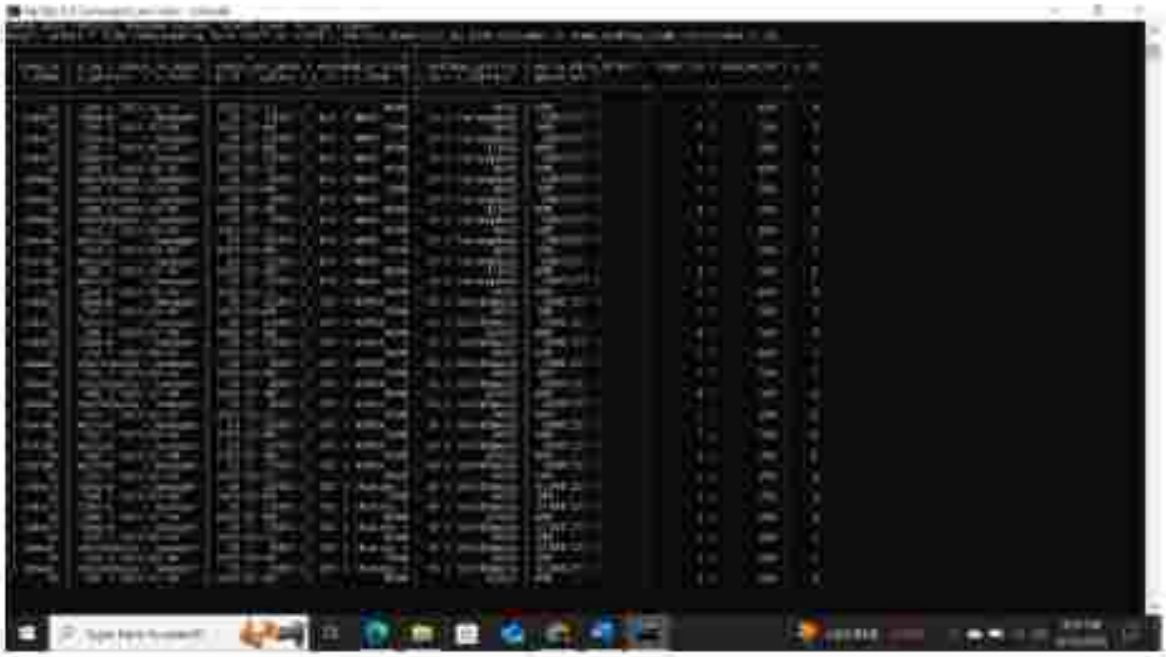
EQUI JOIN

LEFT JOIN

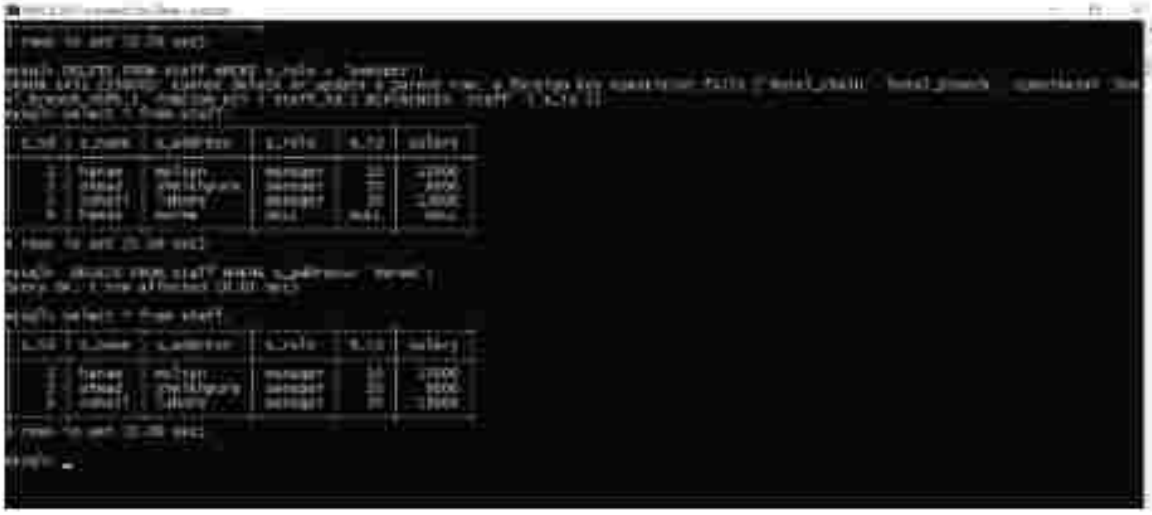


RIGHT JOIN

CROSS JOIN

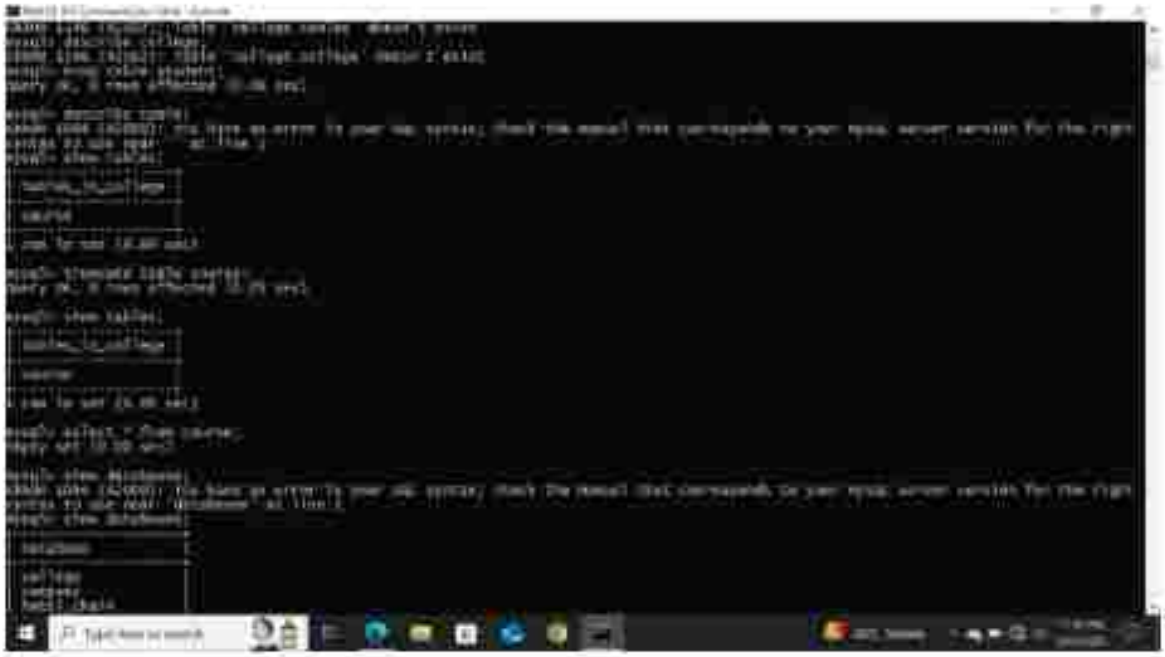


DELETE ROW4 FROM STAFF

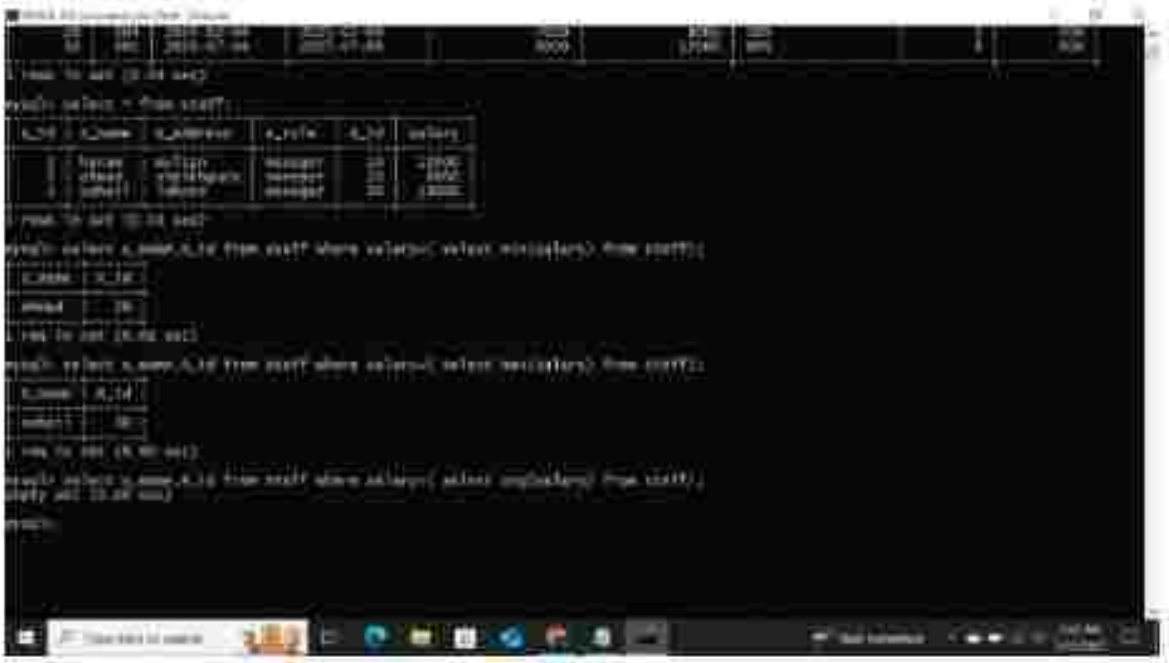


DROP

TRUNCATE



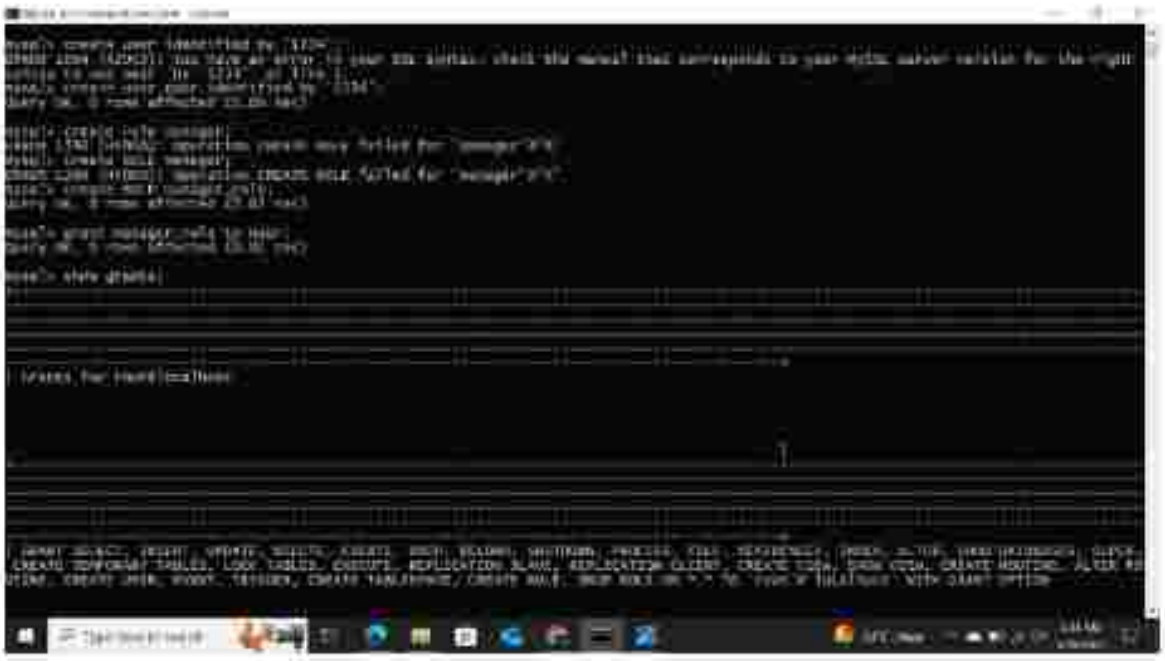
SUBQUERY

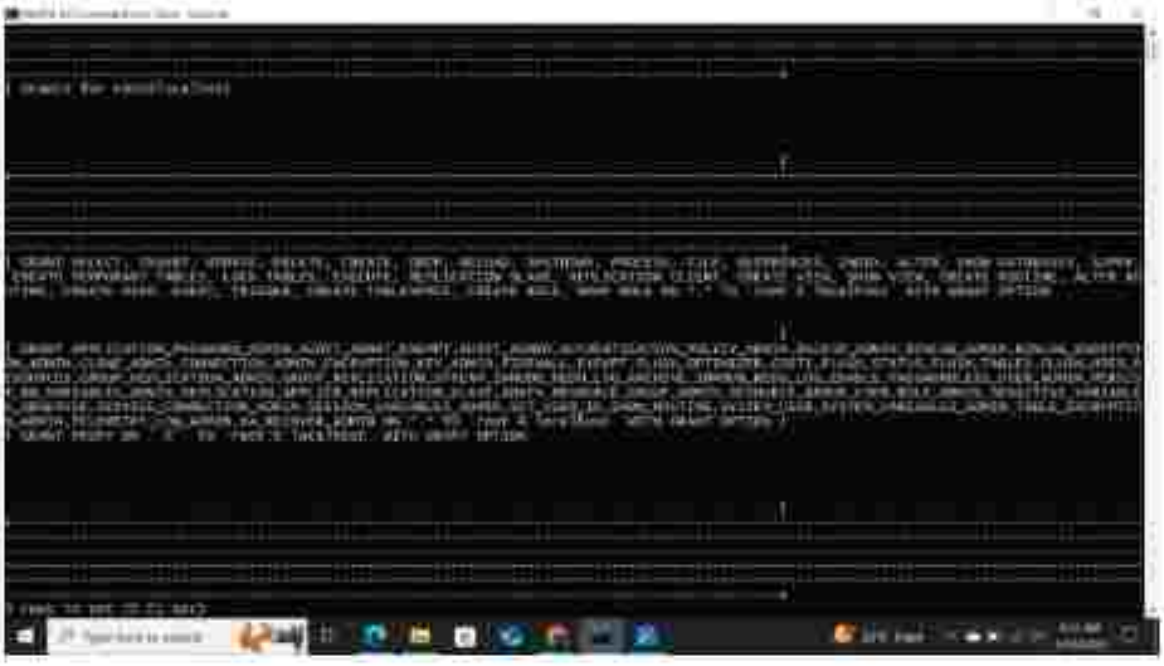


LOWEST SALARY PRINTED

MIN SALARY USING GROUP BY FROM S-ID PRINTED

GRANT





PRIVILEGES