RESTAURANT MANAGEMENT SYSTEM

MINOR PROJECT REPORT

By

PRANAV T(RA2212702010010)
RASHMIYA K (RA2212702010008)

Under the guidance of

Dr. Kanipriya M

In partial fulfilment of the requirements for the degree of

M.TECH INTEGRATED CSE W/S COGNITIVE COMPUTING

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603 203

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Register no. RA2212702010010, RA2212702010008 Certified to be the bonafide work done by **Pranav T, Rashmiya K** of II year/IV sem M.TECH INTEGRATED CSE with Specialization in Cognitive Computing in the Project Course – **21CSC205P Database Management Systems** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur for the academic year 2023-2024.

DATE: 6 May 2024

Faculty in Charge **DEPARTMENT**	**HEAD OF THE** **DEPARTMENT**
Dr. Kanipriya M	Dr. Annie Uthra R
Assistant Professor	Professor & Head
Department of CINTEL	Department of CINTEL
SRMSIT -KTR	SRMSIT -KTR

ABSTRACT

The Restaurant Management System Project in DBMS is a robust software solution designed to streamline and optimize the operations of restaurants and food service establishments. This project aims to provide restaurant owners, managers, and staff with an efficient and intuitive platform for managing reservations, orders, inventory, billing, and customer interactions.

DBMS in restaurant management systems serve as the foundational framework for storing, organizing, and processing data related to menu items, customer preferences, table bookings, staff schedules, inventory levels, and financial transactions. By centralizing data management and providing efficient data retrieval mechanisms, DBMS facilitate smoother restaurant operations, improve customer service, and enable data-driven decision-making.

The advantages of using DBMS in restaurant management systems are diverse. Firstly, DBMS ensure data accuracy and consistency by enforcing data integrity constraints, minimizing errors in order processing and billing. Secondly, DBMS support scalability, allowing restaurants to handle increased customer demand, menu variations, and business expansion without compromising system performance. Thirdly, DBMS enable secure multi-user access, concurrency control, and data privacy measures, ensuring reliable and efficient operations even during peak hours and high transaction volumes.

TABLES OF CONTENT

CHAPTER-1

RESTAURANT MANAGEMENT SYSTEM

ENTITY-RELATIONSHIP DIAGRAM-

Brief Introduction:

An E-R(Entity-Relationship) diagram is a visual representation of the relationships between entities in a database. It's used to design databases by illustrating entities (such as people, objects, or concepts) and the relationships between them.

Creating an Entity-Relationship (ER) diagram for an electricity billing system involves identifying key entities, their attributes, and relationships within the system. Here's a simplified ER diagram for an electricity billing system:

ENTITIES:

1. Advertisements- Attributes: Deadline, Advertisment_Area,Theme,Budget,Customer_ID

2. Bill - Attributes: Bill_ID,Total count,Handled By, Table ID

3. Cashier - Attributes: Admin_ID ,Bill ,_Food Items,Customer Name,Price

4. Chef - Attributes: Gender,DOB,Name,Salary,Chef ID

5. Customer - Attributes: Customer Name,Phone Number,Date,Time,Type ,Customer_ID

6. Employee - Attributes: Gender,Salary,Name,Employee ID

7.Food_Delivery-Attributes:Customer_FID,Total_Price,Items_Details,Delivery Partner,Branch ID

8. Manager - Attributes:Manager_ID,Name,Salary,Gender

9. Menu - Attributes: Timing,Cuisine,Price,Food ID

10. Order - Attributes: Table No,Time,Price,Customer ID

11. Restaurant - Attributes: GST NO,Employee Month,Profit,Address Landmark,Branch ID,Address City

12. Skill - Attributes: Chef Name,Skill 1,Skill 2,Skill 3

13. Vendor - Attributes: Customer_Name,Price,Quantity ,Customer VID,Product Name

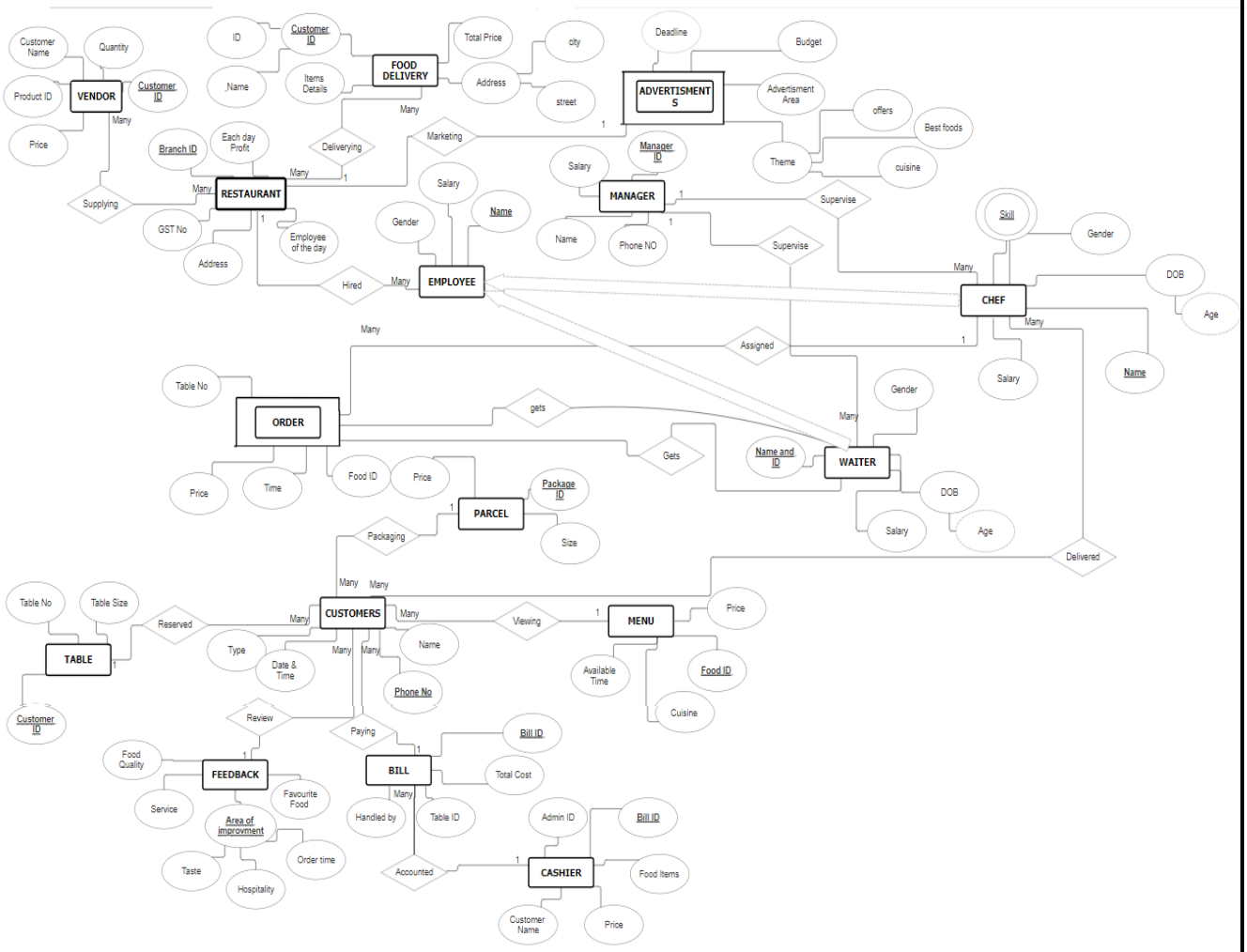
14. Table - Attributes: Table No,Customer ID,Table Size

15. Waiter - Attributes: Name ID,Gender,DOB,Salary,Waiter ID

RELATIONSHIPS

1. 1 Relationship between **ADVERTISMENTS** and **CUSTOMER** entities:
    - Many-to-One relationship, as many advertisements can be linked to one customer.
2. Relationship between **BILL** and **CASHIER** entities:
    - One-to-Many relationship, as one bill can be handled by many cashiers.
3. Relationship between **CASHIER** and **EMPLOYEE** entities:
    - Many-to-One relationship, as many cashiers can be associated with one employee.
4. Relationship between **CHEF** and **EMPLOYEE** entities:
    - One-to-One relationship, as each chef corresponds to one employee.
5. Relationship between **CUSTOMER** and **FEEDBACK** entities:
    - One-to-One relationship, as each customer can provide one feedback.
6. Relationship between **CUSTOMER** and **FOOD_DELIVERY** entities:
    - One-to-Many relationship, as one customer can have many food deliveries.

REPRESENTATION



CHAPTER-2

CONVERTING ER DIAGRAM TO RELATIONAL TABLE

ADVERTISMENTS:

Deadline	Advertisment_Area	Theme	Budget	Customer_ID

BILL:

Bill ID	Total_count	Handled By	Table ID

CASHIER:

Admin_ID	Bill ID	_Food Items	Customer Name	Price

CHEF:

_Gender	DOB_	_Name	Salary	Chef ID

CUSTOMER:

Customer Name	Phone Number	Date Time	Type	Customer ID

EMPLOYEE:

Gender	Salary	Name_	Employee ID

FEEDBACK:

Food Quality	Service	Favorite Food_	Customer	Improvement

FOOD_DELIVERY:

Customer FID	Total Price	Items Details	Delivery Partner	Branch ID

MANAGER:

Manager_ID	Name	Salary	Gender

MENU:

Timing_	Cuisine_	Price_	Food ID_

ORDER_:

Table No	Time	Price	Customer ID

RESTAURANT_:

GST NO	Employee Month	Profit	Address Landmark	Branch ID	Address City

SKILL:

Chef Name	Skill 1	Skill 2	Skill 3

VENDOR:

Customer_Name	Price_	Quantity	Customer VID	Product Name

TABLE1:

Table No	Customer ID	Table Size

WAITER:

Name ID	Gender_	DOB	Salary	Waiter ID

PITFALLS OF RELATIONAL DATABASE SYSTEM:

1. Inadequate Normalization: Normalization is the process of organizing data into tables to minimize redundancy and eliminate data anomalies. Failure to normalise data can result in redundant data and update anomalies. This can lead to data inconsistencies and poor performance.

2.Overuse of NULL values: Using NULL values can make it difficult to query data and can lead to confusion when interpreting data. Overuse of NULL values can also result in poor performance.

3. Poor Indexing: Indexing is essential for efficient querying of data. Poorly designed indexes can result in slow query performance and database bloat.

4. Insufficient Primary and Foreign Keys: Primary and foreign keys establish relationships between tables and ensure data consistency. Failure to implement these keys can result in data inconsistencies and poor performance.

5. Denormalization: While denormalization can improve query performance, it can also lead to data inconsistencies and update anomalies. Denormalization should be used sparingly and only after careful consideration.

6. Failure to Plan for Growth: A database should be designed with future growth in mind. Failure to plan for growth can result in poor performance, data inconsistencies, and costly database redesigns.

7. Lack of Documentation: A lack of documentation can make it difficult to understand the database design and lead to errors in data analysis and reporting.

CREATING TABLES IN THE DATABASE

```
CREATE TABLE ADVERTISMENTS (
  DEADLINE DATE NOT NULL,
  ADVERTISMENT_AREA VARCHAR(50),
  THEME VARCHAR(50),
  BUDGET DECIMAL(10, 0),
  CUSTOMER_ID VARCHAR(20) NOT NULL
);

CREATE TABLE BILL (
  BILL_ID VARCHAR(20) NOT NULL,
  TOTAL_COUNT INT,
  HANDLED_BY VARCHAR(20),
  TABLE_ID INT,
  PRIMARY KEY (BILL_ID)
);

CREATE TABLE CASHIER (
```

```
  ADMIN_ID VARCHAR(20),
  BILL_ID VARCHAR(20) NOT NULL,
  FOOD_ITEMS VARCHAR(50),
  CUSTOMER_NAME VARCHAR(20) NOT NULL,
  PRICE DECIMAL(10, 0),
  PRIMARY KEY (CUSTOMER_NAME)
);

CREATE TABLE CHEF (
  GENDER VARCHAR(20),
  DOB DATE,
  NAME VARCHAR(20) NOT NULL,
  SALARY VARCHAR(20),
  CHEF_ID VARCHAR(20),
  PRIMARY KEY (NAME),
  FOREIGN KEY (CHEF_ID) REFERENCES EMPLOYEE (EMPLOYEE_ID)
);

CREATE TABLE CUSTOMER (
  CUSTOMER_NAME VARCHAR(20),
  PHONE_NO VARCHAR(20),
  DATE_TIME VARCHAR(20),
  TYPE VARCHAR(20),
  CUSTOMER_ID VARCHAR(20) NOT NULL,
  PRIMARY KEY (CUSTOMER_ID)
);

CREATE TABLE EMPLOYEE (
  GENDER VARCHAR(20),
  SALARY DECIMAL(10, 0),
  NAME VARCHAR(20),
  EMPLOYEE_ID VARCHAR(20) NOT NULL,
  PRIMARY KEY (EMPLOYEE_ID)
);

CREATE TABLE FEEDBACK (
  FOOD_QUALITY VARCHAR(20),
  SERVICE VARCHAR(50),
  FAVOURITE_FOOD VARCHAR(30),
  IMPROVEMENT_TASTE VARCHAR(100) NOT NULL,
  IMPROVEMENT_HOSPITALITY VARCHAR(100),
  IMPROVEMENT_ORDER_TIME VARCHAR(100),
  CUSTOME_ID VARCHAR(20) NOT NULL,
  PRIMARY KEY (IMPROVEMENT_TASTE)
);
```

```
CREATE TABLE FOOD_DELIVERY (
  CUSTOMER_FID VARCHAR(20),
  TOTAL_PRICE DECIMAL(10, 0),
  ITEMS_DETAILS VARCHAR(100),
  DELIVERY_PARTNER VARCHAR(40),
  BRANCH_ID VARCHAR(20)
);
ALTER TABLE FOOD_DELIVERY ADD CONSTRAINT
FOOD_DELIVERY_FK1 FOREIGN KEY (BRANCH_ID) REFERENCES
RESTAURANT_ (BRANCH_ID);

CREATE TABLE MANAGER (
  MANAGER_ID VARCHAR(20),
  NAME VARCHAR(20) NOT NULL,
  SALARY DECIMAL(10, 0),
  GENDER VARCHAR(20),
  PRIMARY KEY (NAME),
  FOREIGN KEY (MANAGER_ID) REFERENCES EMPLOYEE (EMPLOYEE_ID)
);

CREATE TABLE MENU (
  TIMING VARCHAR(20),
  CUISINE VARCHAR(20),
  PRICE DECIMAL(10, 0),
  FOOD_ID VARCHAR(20) NOT NULL,
  PRIMARY KEY (FOOD_ID)
);

CREATE TABLE ORDER_ (
  TABLE_NO INT,
  TIME VARCHAR(20),
  PRICE DECIMAL(10, 0),
  CUSTOMER_ID VARCHAR(20)
);

CREATE TABLE PARCEL (
  PRICE DECIMAL(10, 0),
  PACKAGE_ID VARCHAR(20),
  QUANTITY VARCHAR(20)
);

CREATE TABLE RESTAURANT_ (
  PROFIT DECIMAL,
  GST_NO VARCHAR(20),
  EMPLOYEE_MONTH VARCHAR(20),
  ADDRESS_LANDMARK VARCHAR(100),
```

```
  BRANCH_ID VARCHAR(20) NOT NULL,
  ADDRESS_CITY VARCHAR(20),
  PRIMARY KEY (BRANCH_ID)
);

CREATE TABLE SKILL (
  CHEF_NAME VARCHAR(30),
  SKILL1 VARCHAR(30),
  SKILL2 VARCHAR(30),
  SKILL3 VARCHAR(30)
);
ALTER TABLE SKILL ADD CONSTRAINT SKILL_FK1 FOREIGN KEY
(CHEF_NAME) REFERENCES CHEF (NAME);

CREATE TABLE TABLE1 (
  TABLE_NO INT,
  CUSTOMER_ID VARCHAR(20),
  TABLE_SIZE INT
);
ALTER TABLE TABLE1 ADD CONSTRAINT TABLE1_FK1 FOREIGN KEY
(CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID);

CREATE TABLE VENDOR (
  CUSTOMER_NAME VARCHAR(20),
  PRICE DECIMAL(10, 0),
  QUANTITY INT,
  CUSTOMER_VID VARCHAR(20),
  PRODUCT_NAME VARCHAR(20)
);
ALTER TABLE VENDOR ADD CONSTRAINT VENDOR_FK1 FOREIGN KEY
(CUSTOMER_VID) REFERENCES RESTAURANT_ (BRANCH_ID);

CREATE TABLE WAITER (
  NAME_ID VARCHAR(20) NOT NULL,
  GENDER VARCHAR(20),
  DOB DATE,
  SALARY VARCHAR(20),
  WAITER_ID VARCHAR(20),
  PRIMARY KEY (NAME_ID),
  FOREIGN KEY (WAITER_ID) REFERENCES EMPLOYEE (EMPLOYEE_ID)
);
```

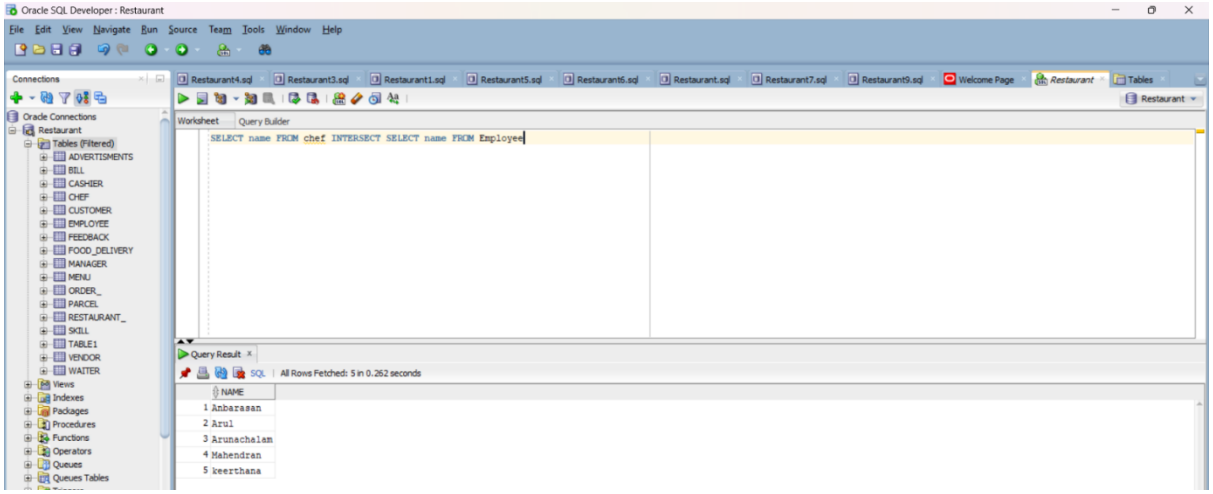
CHAPTER-3

QUERIES

1.Subquery to Get Chef Name from Employees List:

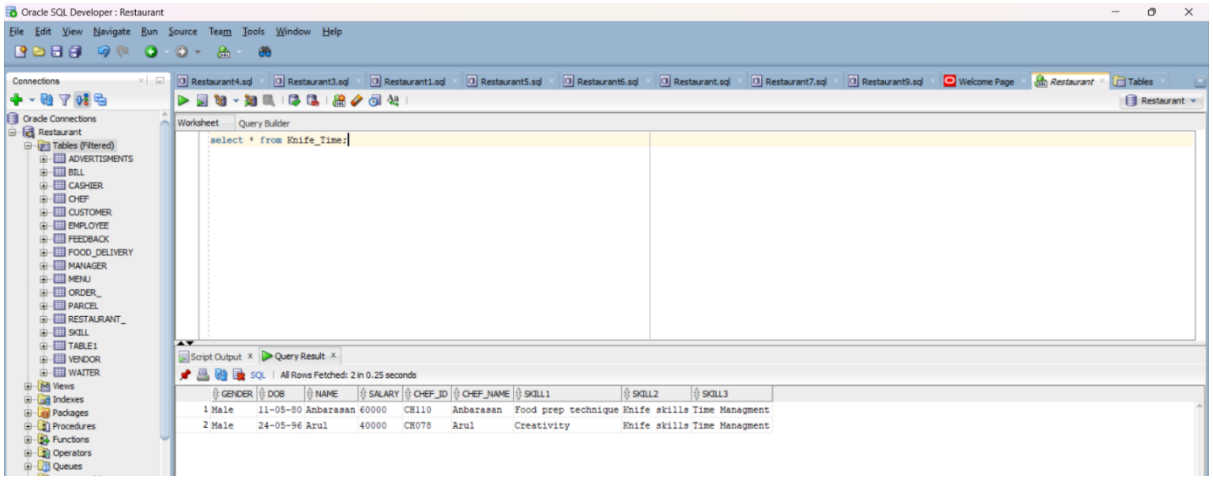
=> SELECT name FROM chef INTERSECT SELECT name

FROM Employee



2.Subquery to Get Chef Skills as view table:

=> create view Knife_Time as (SELECT * FROM chef r JOIN skill a ON a.chef_name = r.name and skill2='Knife skills' and skill3='Time Managment');



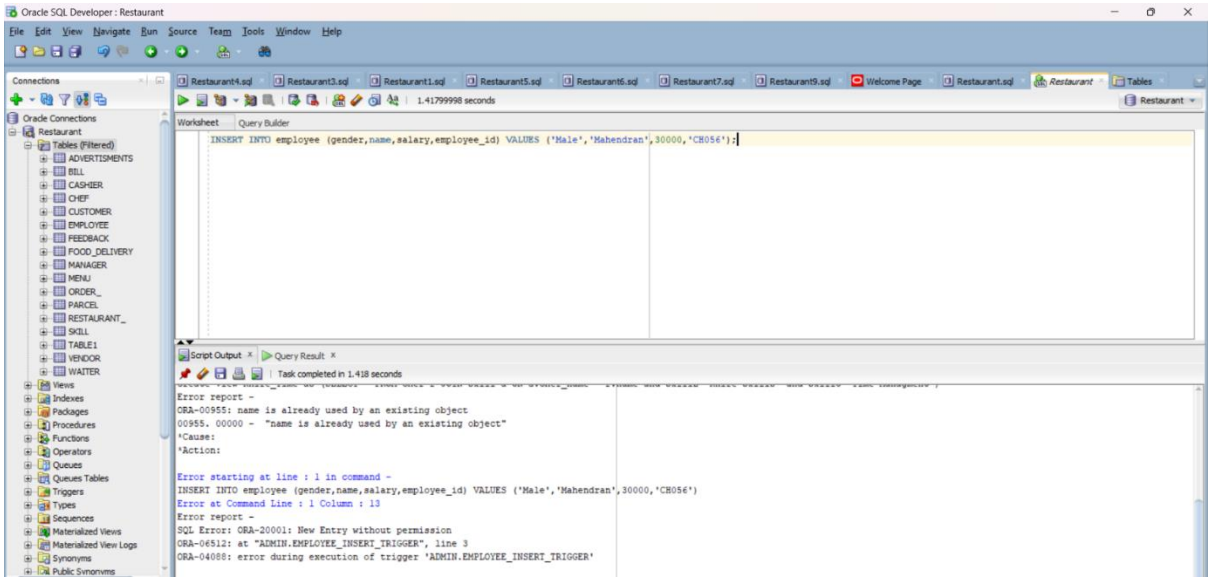
3.Creating a query in order to avoid inconsistent area:

=> CREATE OR REPLACE TRIGGER customer_before_insert_trigger

BEFORE INSERT ON chef

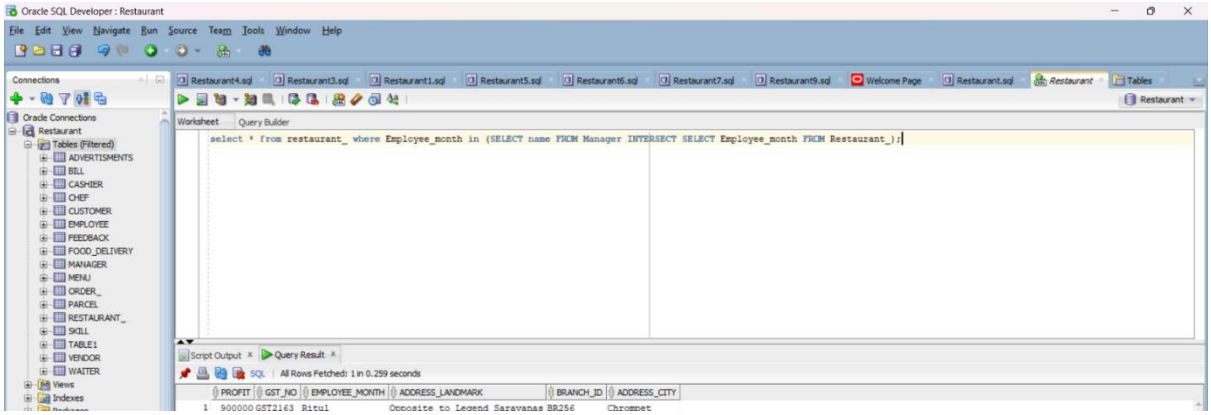
FOR EACH ROW

```
        BEGIN
          IF :NEW.GENDER IS NULL THEN
            RAISE_APPLICATION_ERROR(-20001, 'Chef Gender cannot be null');
  END IF;
END;
```



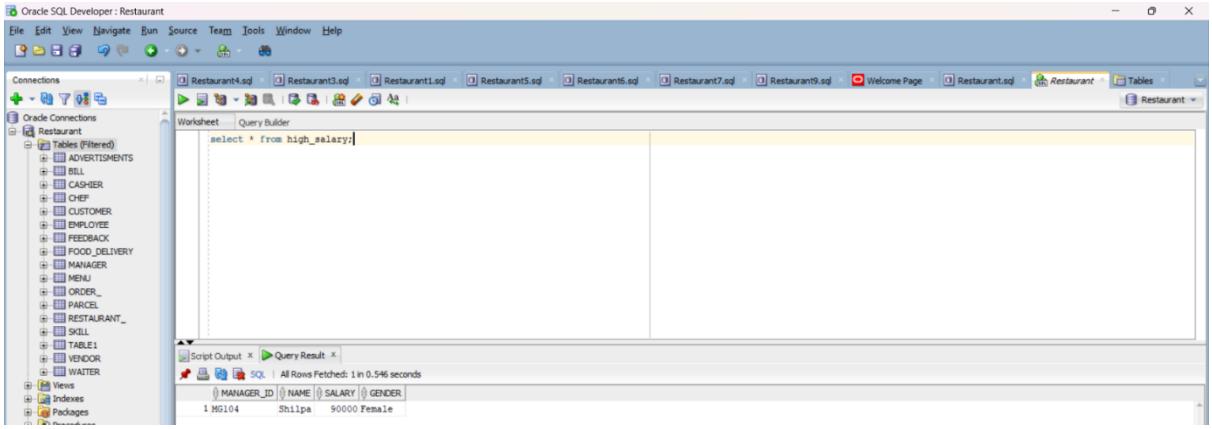
4. Subquery to Get Employee of the month:

=> SELECT * from restaurant_ where Employee_month in (SELECT name FROM Manager INTERSECT SELECT Employee_month FROM Restaurant_);



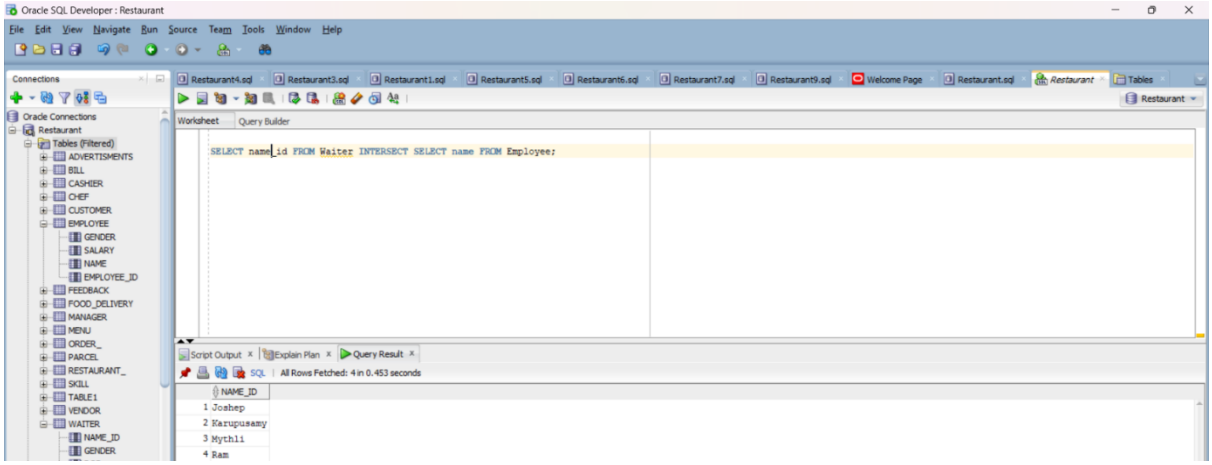
5.Subquery to Get the Top Paid Salary:

=> CREATE VIEW High_salary AS SELECT * FROM manager WHERE salary = (SELECT MAX(salary) FROM manager);



6.Subquery to Get The waiters who are handling the bill:

=> SELECT waiter_id FROM Waiter INTERSECT SELECT name FROM Employee;



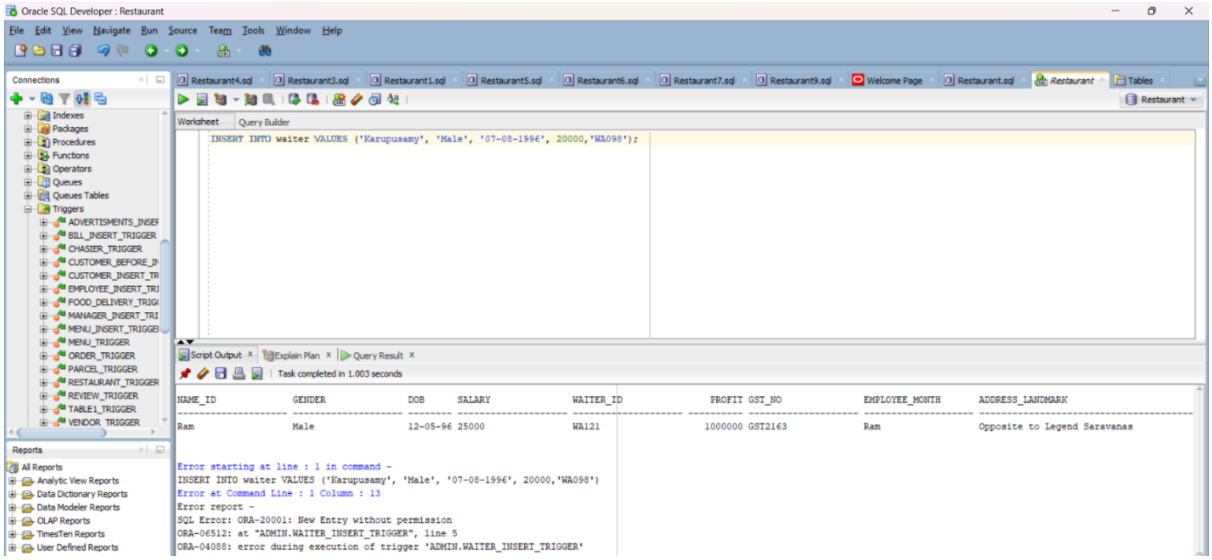
7.Creating a trigger to avoid alteration of salary

=> CREATE OR REPLACE TRIGGER Waiter_insert_trigger

BEFORE INSERT ON waiter
FOR EACH ROW
DECLARE
max_salary NUMBER;
BEGIN
SELECT MAX(salary) INTO max_salary FROM waiter;

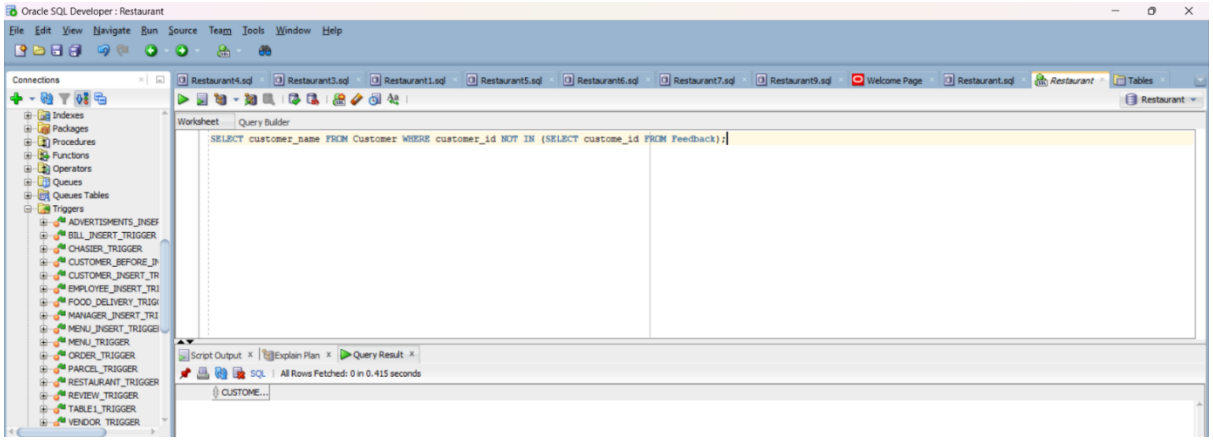
IF :NEW.salary > max_salary THEN
RAISE_APPLICATION_ERROR(-20001, 'Highest salary is being entered');
END IF;

END;



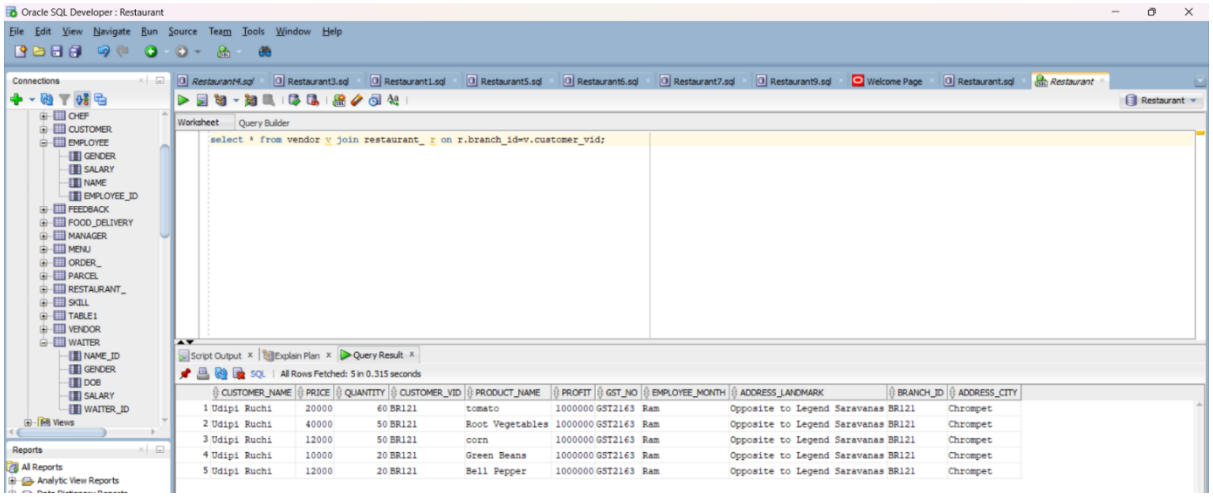
9.Checking out the customers data not existing:

=> SELECT customer_name FROM Customer WHERE customer_id NOT IN (SELECT custome_id FROM Feedback);



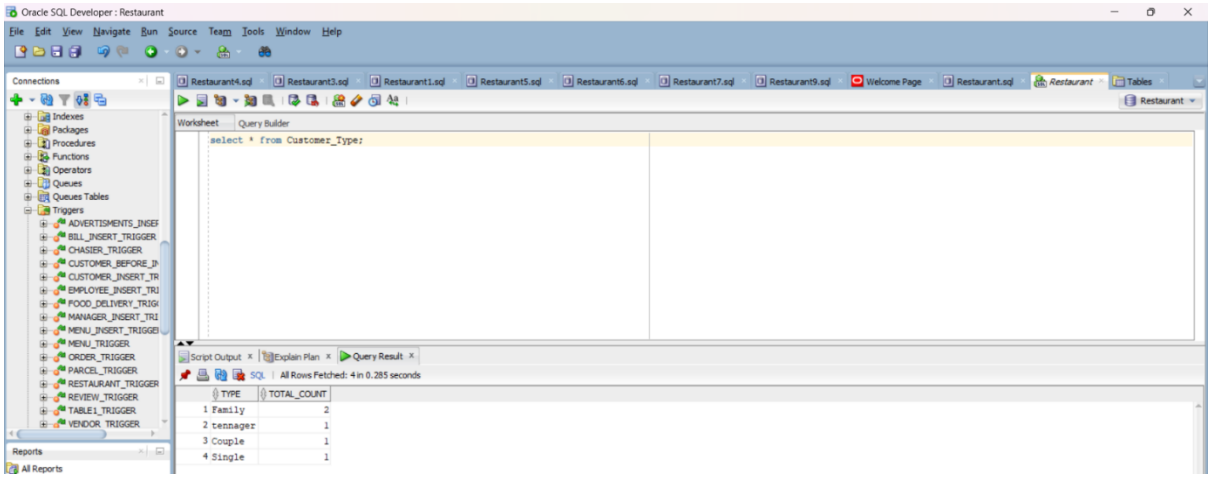
10.Join to Get Customer Name with FeedBack Details:

=> SELECT d.*,f.* FROM customer d join feedback f on d.customer_id=f.custome_id;



11.Creating View Table and grouping them based on their type:

=> CREATE view Customer_Type as (select Type,Count(Type) as Total_count from Customer group by Type);



12.Trigger to avoid new Entry without permission:

=> BEFORE INSERT ON customer

FOR EACH ROW

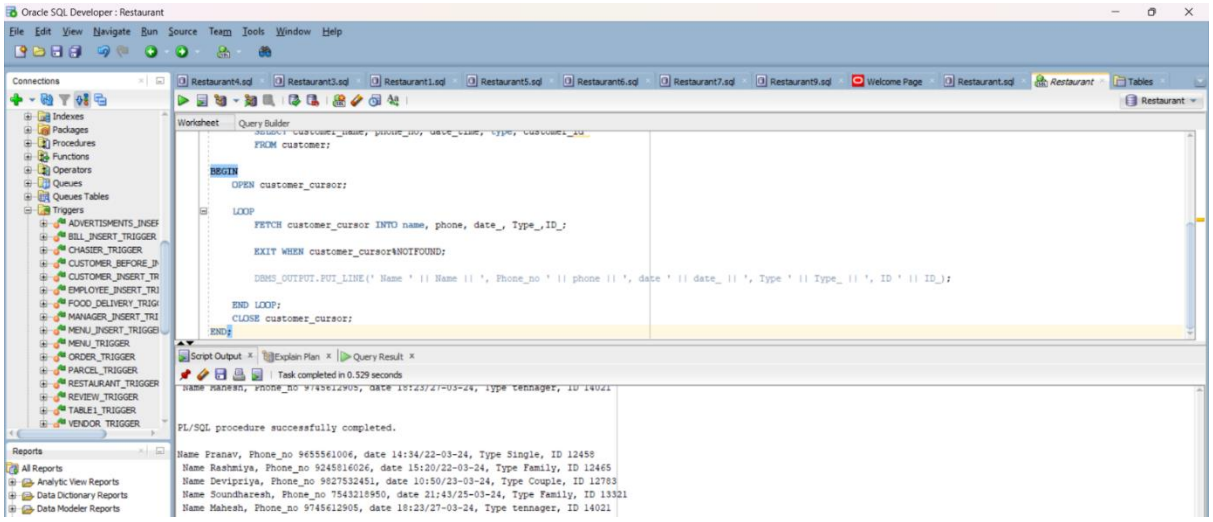
BEGIN

IF :NEW.customer_id is NOT NULL THEN

RAISE_APPLICATION_ERROR(-20001, 'New Entry without permission');

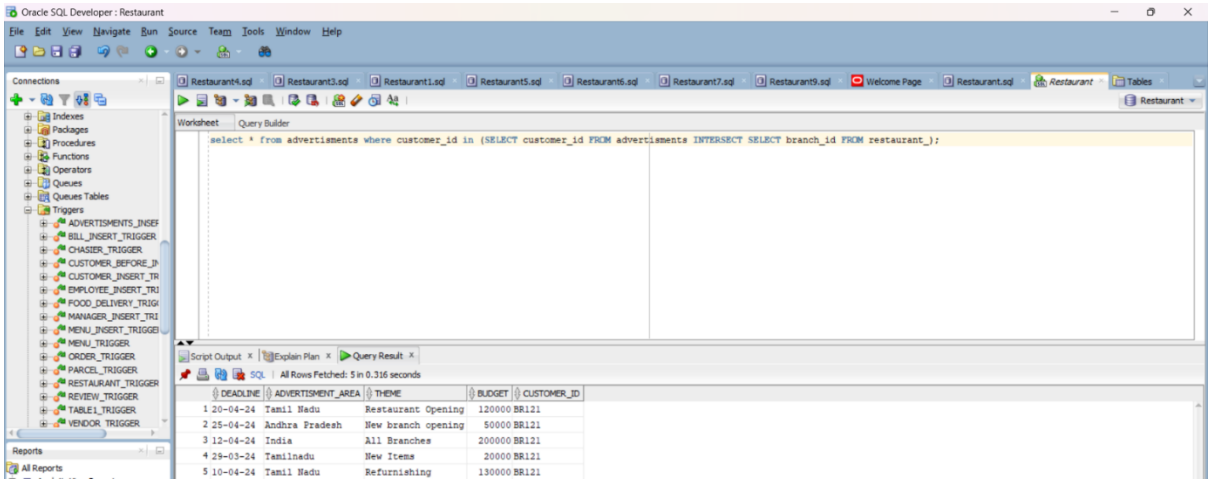
END IF;

END;



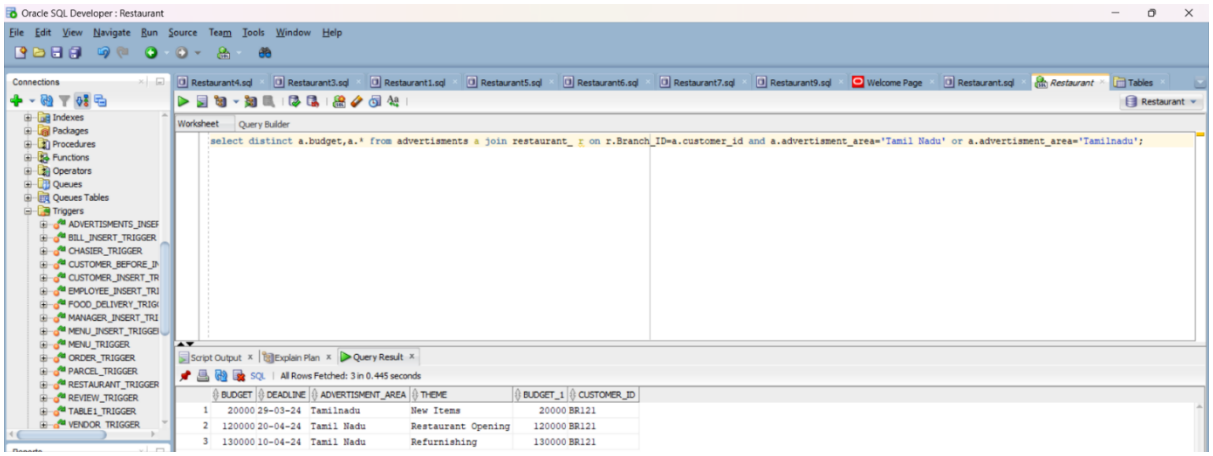
13. Subquery to get advertisement details based on branch_id:

=> SELECT * from advertisments where customer_id in (SELECT customer_id FROM advertisments INTERSECT SELECT branch_id FROM restaurant_);



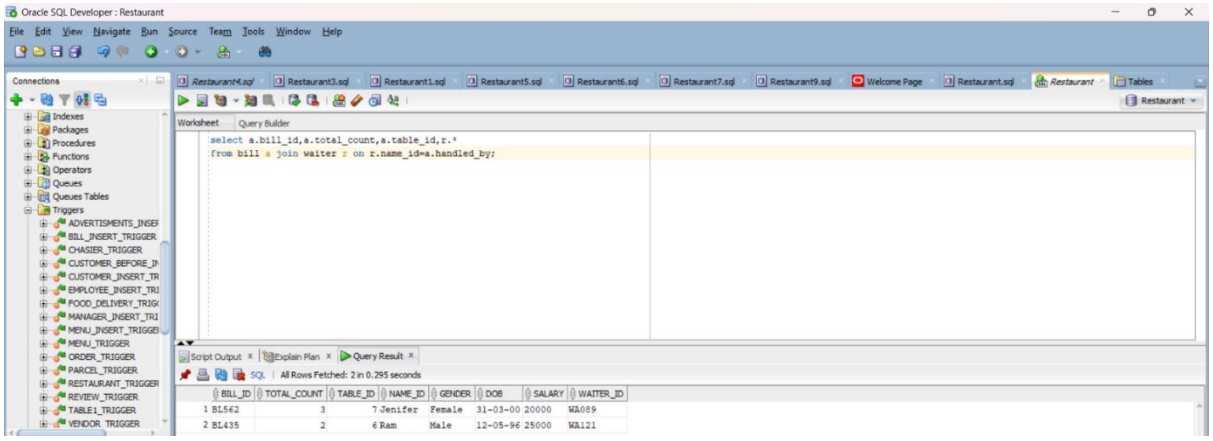
14. Join to get advertisement details based on Location

=> SELECT distinct a.budget,a.* from advertisments a join restaurant_ r on r.Branch_ID= a.customer_id and a.advertisment_area='Tamil Nadu' or a.advertisment_area='Tamilnadu';



15.Join to get bill and waiter together:

=> SELECT a.bill_id,a.total_count,a.table_id,r.* from bill a join waiter r on r.name_id= a.handled_by;



16.Trigger to avoid new entry in bill system without permission

=> CREATE OR REPLACE TRIGGER bill_insert_trigger

BEFORE INSERT ON bill

FOR EACH ROW

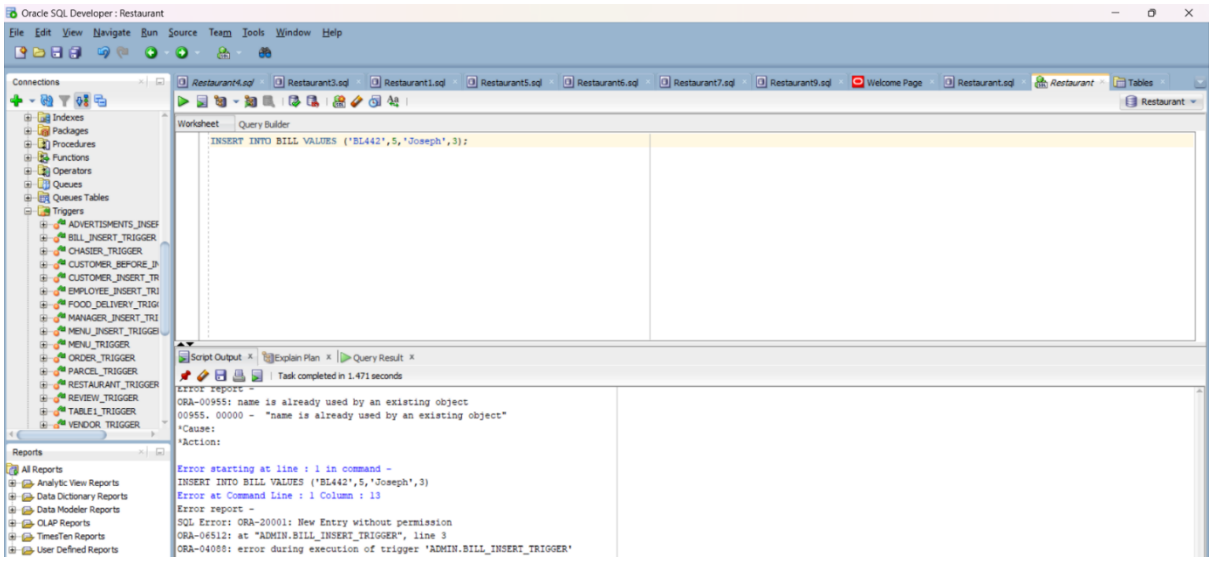
BEGIN

IF :NEW.bill_id is NOT NULL THEN

RAISE_APPLICATION_ERROR(-20001, 'New Entry without permission');

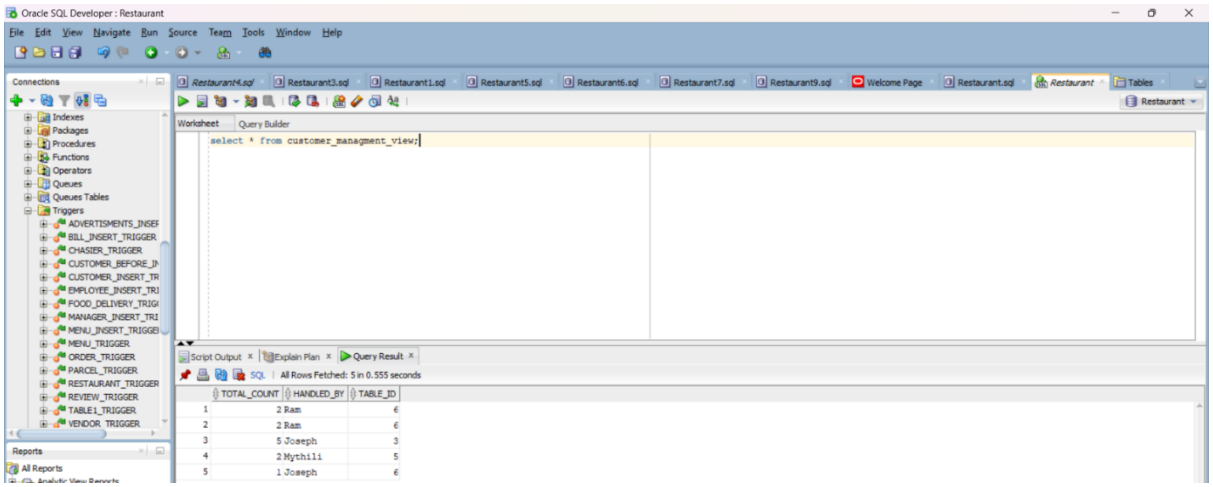
END IF;

END;



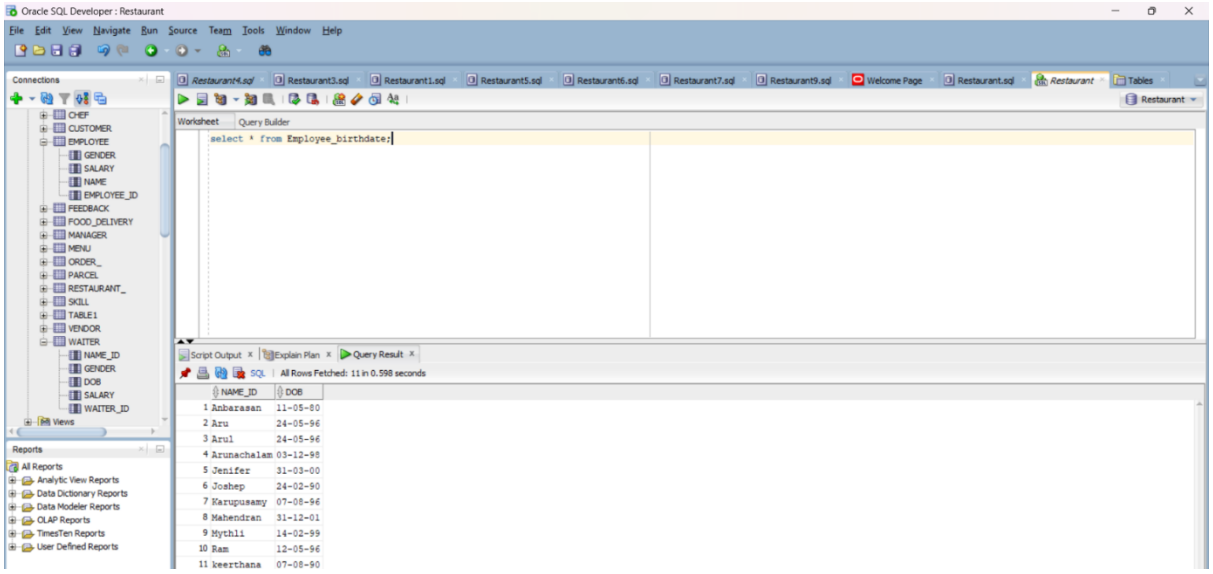
17.Subquery to create a customer_manager view table:

=> CREATE VIEW customer_managment_view as (select total_count,handled_by,table_id from bill b join cashier c on c.bill_id=b.bill_id);



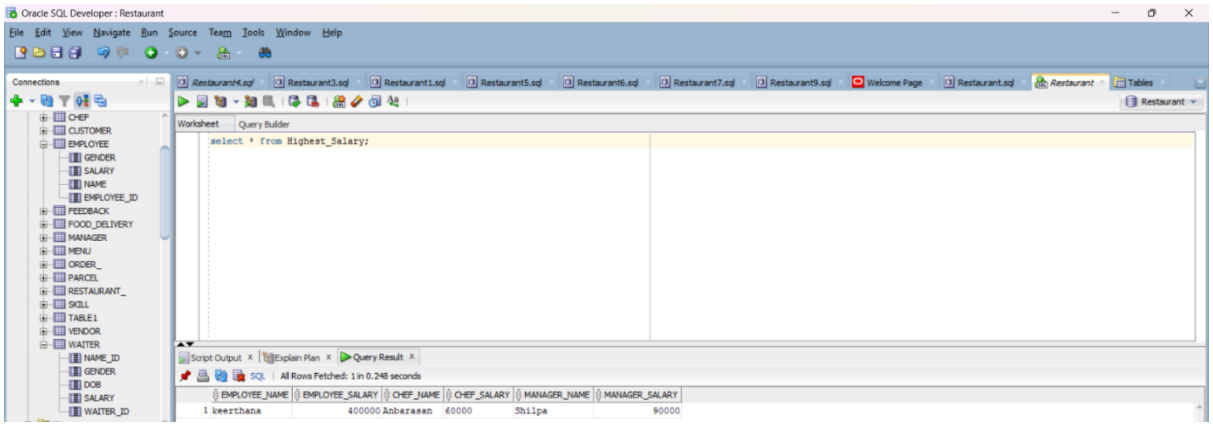
18. What are the Birthdate of all the employees?

=> CREATE VIEW Employee_birthdate as (select Name_ID,DOB from WAITER union select Name,DOB from Chef);



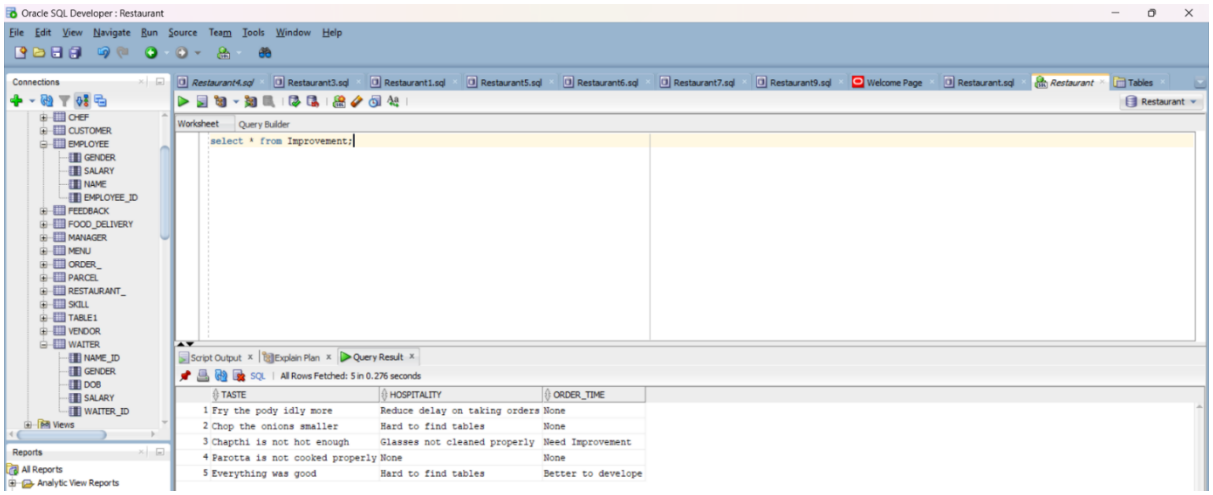
19.Create a view table to find all the highest salary is different sections:

=> CREATE VIEW Highest_Salary AS SELECT E.Name AS Employee_Name, E.Salary AS Employee_Salary, C.Name AS Chef_Name, C.Salary AS Chef_Salary, M.Name AS Manager_Name, M.Salary AS Manager_Salary FROM Employee E, Chef C, Manager M WHERE E.Salary = (SELECT MAX(Salary) FROM Employee) AND C.Salary = (SELECT MAX(Salary) FROM Chef) AND M.Salary = (SELECT MAX(Salary) FROM Manager);



20. To get the details of the improvements given by customers in feedback:

=> CREATE VIEW Improvement as (select Improvement_Taste as Taste, Improvement_Hospitality as Hospitality, Improvement_Order_Time as Order_Time from Feedback);



21. Trigger to view the feedback while entering:

=> CREATE OR REPLACE TRIGGER Review_trigger

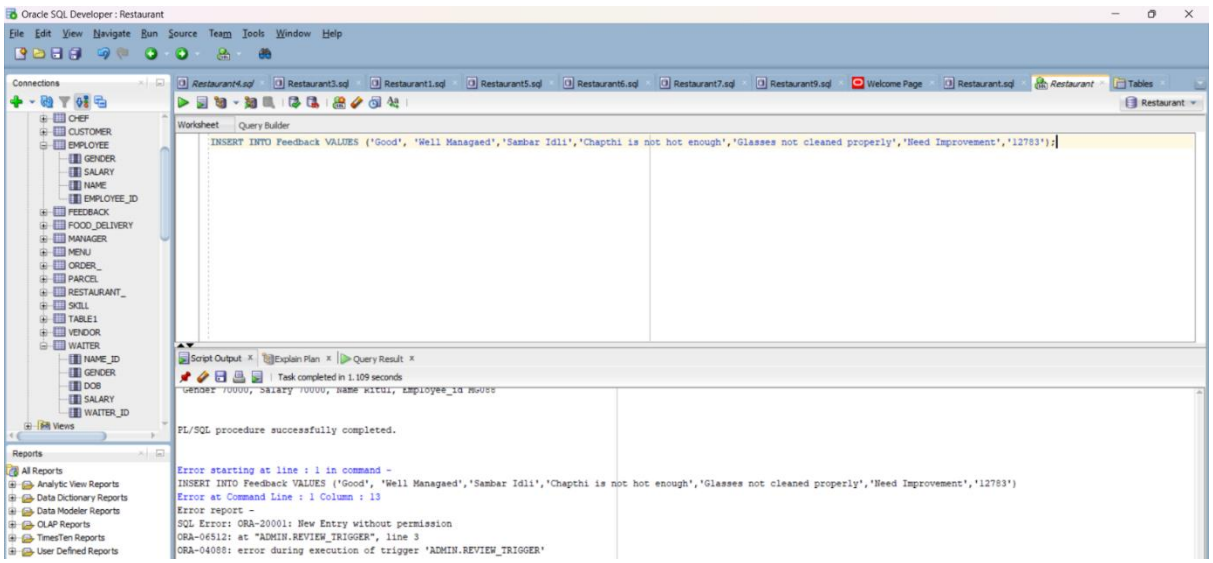
AFTER INSERT ON Feedback

FOR EACH ROW

BEGI'

DBMS_OUTPUT.PUT_LINE('Food Quality' || :new.Food_quality || 'Service' || :new.Service || 'Favourite' || :new.Favourite || ' Ref customer ID ' || :new.customer_id);

END;

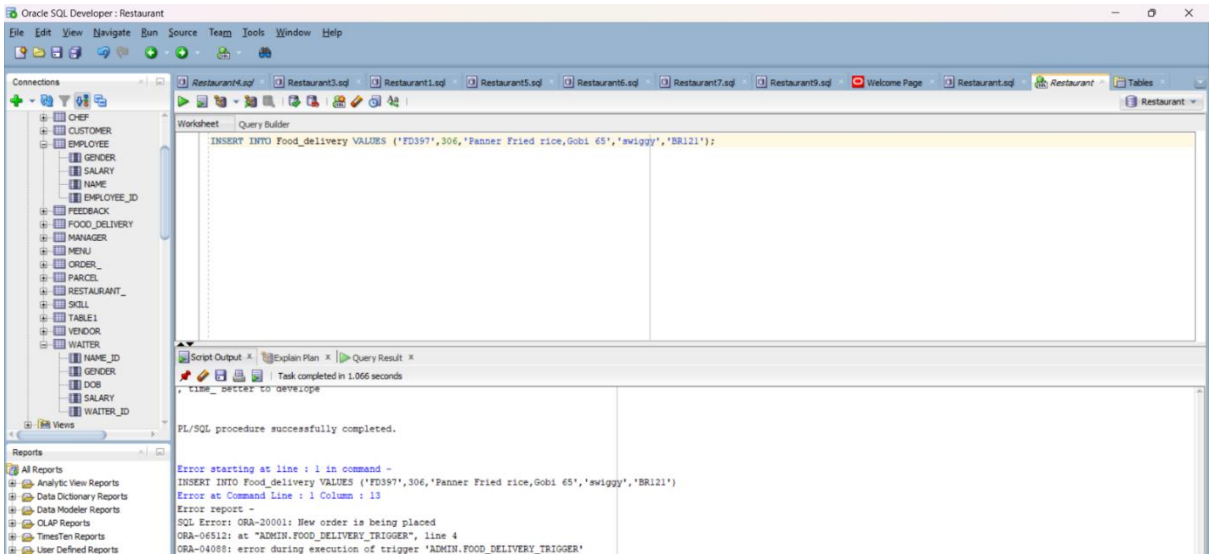


22. Trigger for Automatically Generating Complaints in food Delivery:

=> CREATE OR REPLACE TRIGGER food_delivery_trigger

AFTER INSERT ON food_delivery

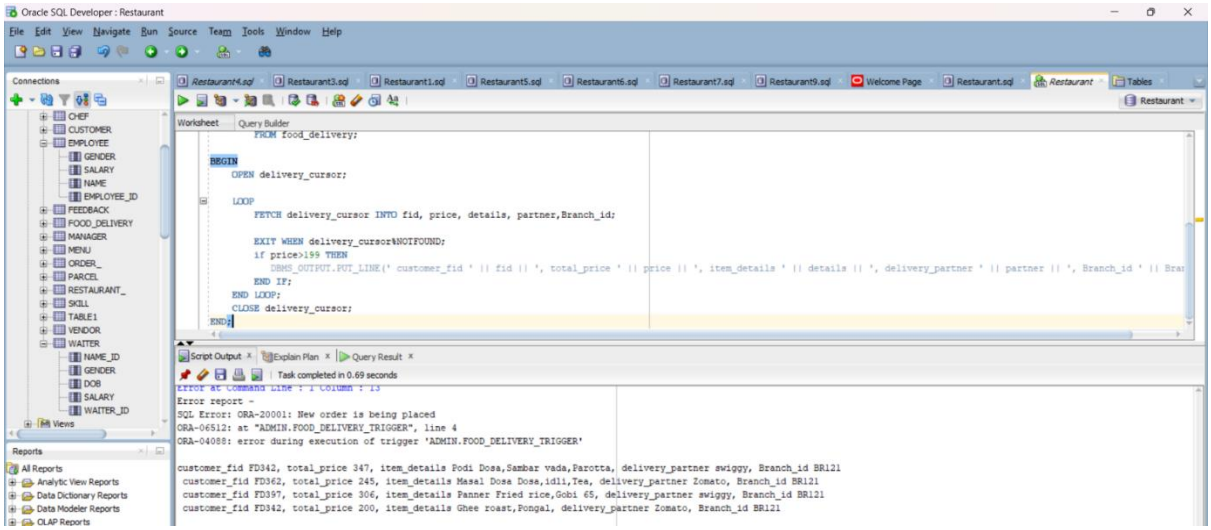
```
FOR EACH ROW
BEGIN

   IF :NEW.customer_fid is NOT NULL THEN
      RAISE_APPLICATION_ERROR(-20001, 'New order is being placed');
   END IF;
END;
```



23. Is this PL/SQL code designed to efficiently retrieve and display information about Food Delivery?

```
SET SERVEROUTP  ON;

DECLARE

   fid food_delivery.customer_fid%TYPE;

   price food_delivery.total_price%TYPE;

   details food_delivery.items_details%TYPE;

   partner food_delivery.delivery_partner%TYPE;

   Branch_id food_delivery.Branch_id%TYPE;

   CURSOR delivery_cursor IS

      SELECT customer_fid, total_price, items_details, delivery_partner,Branch_id
```

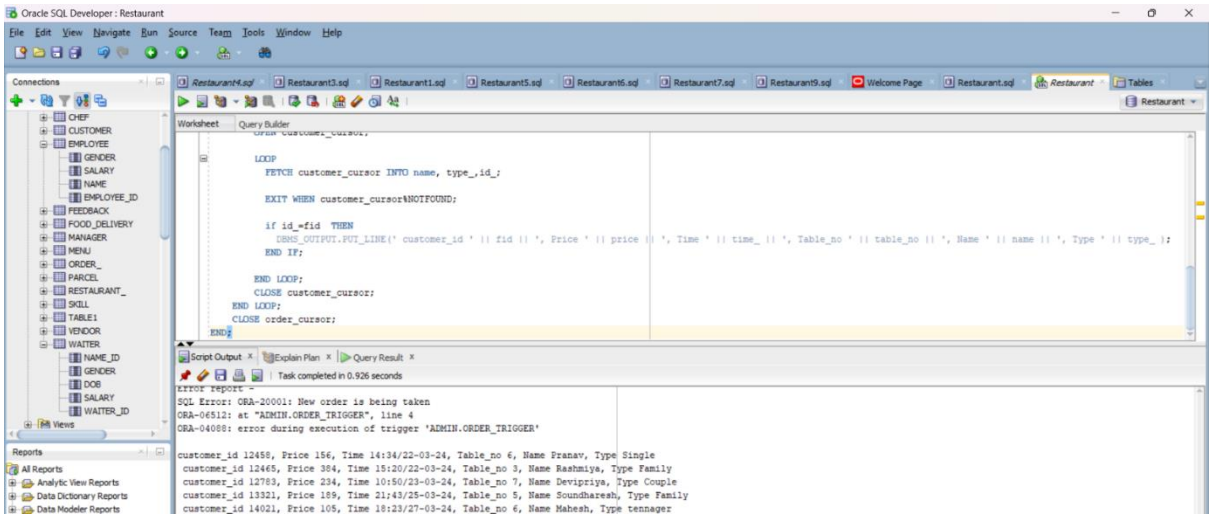
```
    FROM food_delivery;


BEGIN
   OPEN delivery_cursor;


   LOOP
      FETCH delivery_cursor INTO fid, price, details, partner,Branch_id;


      EXIT WHEN delivery_cursor%NOTFOUND;
      if price>199 THEN
        DBMS_OUTPUT.PUT_LINE(' customer_fid ' || fid || ', total_price ' || price || ',
item_details ' || details || ', delivery_partner ' || partner || ', Branch_id ' || Branch_id);
      END IF;
   END LOOP;
   CLOSE delivery_cursor;
END;
```



24. Is this PL/SQL code designed to efficiently retrieve and display information about Order by customers?

```
=> DECLARE
   fid order_.customer_id%TYPE;
   price order_.price%TYPE;
   time_ order_.time%TYPE;
   table_no order_.table_no%TYPE;
   name customer.customer_name%TYPE;
   type_ customer.type%TYPE;
   id_ customer.customer_id%TYPE;


   CURSOR order_cursor IS
      SELECT customer_id, price, time, table_no
      FROM order_;
   CURSOR customer_cursor IS
      SELECT customer_name, type,customer_id
      FROM customer;

BEGIN
   OPEN order_cursor;

   LOOP
      FETCH order_cursor INTO fid, price, time_, table_no;

      EXIT WHEN order_cursor%NOTFOUND;

      OPEN customer_cursor;

      LOOP
        FETCH customer_cursor INTO name, type_,id_;

        EXIT WHEN customer_cursor%NOTFOUND;
```

```
        if id_=fid  THEN
          DBMS_OUTPUT.PUT_LINE(' customer_id ' || fid || ', Price ' || price || ', Time ' || time_
|| ', Table_no ' || table_no || ', Name ' || name || ', Type ' || type_ );
        END IF;

      END LOOP;
      CLOSE customer_cursor;
   END LOOP;
   CLOSE order_cursor;
END;
```



CHAPTER -4

NORMALIZATION

Normalization is the process to eliminate data redundancy and enhance data integrity in the table. Normalization also helps to organize the data in the database. It is a multi-step process that sets the data into tabular form and removes the duplicated data from the relational tables.

TYPES OF NORMAL FORMS IN NORMALIZATION-

1. FIRST NORMAL FORM
2. SECOND NORMAL FORM
3. THIRD NORMAL FORM
4. BOYCE- CODD NORMAL FORM
5. FOURTH NORMAL FORM
6. FIFTH NORMAL FORM

CHEF:

3NF

Functional Dependency

Chef Name → Skill 1, Skill 2, Skill 3

Chef ID → Gender, Date of Birth, Name, Salary

Name → Gender, Date of Birth, Salary, Chef ID

Chef ID → Salary → Gender(transitive dependecy)

It is currently in 2nf because it has a transitive dependency of chef id→salary→Gender. So to convert it into 3nf I have to remove the transitive dependency.

TO CHANGE:

```
1. CREATE TABLE CHEF1 (
  DOB DATE,
  NAME VARCHAR2(20) NOT NULL,
  SALARY VARCHAR2(20),
```

CHEF_ID VARCHAR2(20),

)

2. CREATE TABLE CHEF2 (

GENDER VARCHAR2(20),

CHEF_ID VARCHAR2(20),

)

FOOD DELIVERY:

NO NF

Function Dependency

Customer FID → Total Price, Item Details, Delivery_Partner, Branch ID

Normalization

There is atomicity in this table so we have first solve that multivalued attribute

TO CHANGE:

CREATE TABLE FOOD_DELIVERY1 (

CUSTOMER_FID VARCHAR2(20),

TOTAL_PRICE NUMBER,

ITEMS_DETAILS1 VARCHAR2(100),

ITEMS_DETAILS2 VARCHAR2(100),

ITEMS_DETAILS3 VARCHAR2(100),

DELIVERY_PARTNER VARCHAR2(40),

BRANCH_ID VARCHAR2(20)

);

2NF

Function Dependency

Customer FID → Total Price, Item Details1, Item Details2, Item Details3,Delivery_Partner, Branch ID

Item Details1→ Total Price, Customer FID, Item Details2, Item Details3,Delivery_Partner, Branch ID

Item Details2→ Total Price, Item Details1, Customer FID, Item Details3,Delivery_Partner, Branch ID

Normalization

The is no Partial Primary key dependency so This satisfy the condition for 2nf

3NF

Function Dependency

Customer FID → Total Price, Item Details1, Item Details2, Item Details3,Delivery_Partner, Branch ID

Item Details1→ Total Price, Customer FID, Item Details2, Item Details3,Delivery_Partner, Branch ID

Item Details2→ Total Price, Item Details1, Customer FID, Item Details3,Delivery_Partner, Branch ID

Normalization

There is no Transitive Dependency and Every candidate key is a super key so it satisfy the Boyce code and 4nf

MENU:

5NF

Functional Dependency

Food_ID → Time,Cuisine,Price

Cuisine → Time, Food_ID,Price

Normalization

Its already existing in 3nf form so no need to make any changes in the database.

BILL:

5NF

Functional Dependency

Bill ID→ Quantity, Handled by, Table no

Handled By → Quantity, Bill Number, Table no

Normalization

Its already existing in 3nf form so no need to make any changes in the database.

PITFALLS IN NORMALIZATION CONCEPT

While the provided data appears to be structured and normalized up to at least Second Normal Form (2NF), there are still potential pitfalls and areas for improvement in terms of database design and normalization concepts. Here are some pitfalls and considerations:

1. **Redundancy in Address and Contact Information:**

- In several tables (e.g., User, Customer, ServiceProvider, Employee), there are columns for storing Address and ContactNumber. This can lead to redundancy if the same address or contact number needs to be updated in multiple places. One solution could be to create separate tables for Address and Contact information and link them using foreign keys.

2. **Denormalization for Performance**:

- While normalization helps in reducing redundancy and maintaining data integrity, in some cases, denormalization might be necessary for performance optimization. For example, in a

high-transaction system, joining multiple tables frequently could impact performance. In such cases, carefully denormalizing certain tables or using materialized views can be considered.

3. **Potential Update Anomalies**:

- Update anomalies can occur when data needs to be updated in multiple places, leading to inconsistencies if not handled properly. For instance, if a customer's contact number changes, it needs to be updated in multiple tables (e.g., Customer, User) where it's stored, increasing the risk of inconsistencies.

4**. Lack of Data Validation:**

- Data validation is crucial to ensure data integrity. Without proper validation rules and constraints, the database may accept invalid or inconsistent data, leading to issues in data quality. Implementing data validation checks at the database level can mitigate this risk.

5. **Overly Nested Relationships:**

- While relationships between tables are necessary, overly nested relationships can make queries complex and impact performance. It's essential to strike a balance between maintaining relationships for data integrity and optimizing query performance.

6. **Incomplete Normalization**:

- Although the provided data seems to be normalized up to 2NF, further normalization (e.g., Third Normal Form - 3NF) could be beneficial in some cases. Analyzing functional dependencies and eliminating transitive dependencies can lead to a more robust and efficient database design.

7. **Handling Historical Data:**

- If historical data tracking is required (e.g., tracking changes in meter readings over time), additional considerations for data storage and retrieval mechanisms may be needed. Implementing effective techniques such as versioning or audit trails can address this requirement without compromising normalization.

8. **Optimizing Indexing and Query Performance**:

- While normalization focuses on data organization, indexing and optimizing queries are essential for efficient data retrieval. Proper indexing strategies, query optimization techniques,

and understanding the database engine's capabilities are crucial for improving overall system performance.

Addressing these pitfalls involves a combination of thoughtful database design, adherence to normalization principles, implementing data validation rules, optimizing performance, and considering specific business requirements for data storage and retrieval.

CHAPTER-5

Implementation of concurrency control and recovery mechanisms

```
mysql> use restaurant
Database changed
mysql> LOCK TABLES WAITER WRITE, EMPLOYEE WRITE,TRANSACTION_LOG WRITE;
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> UPDATE WAITER SET SALARY = '30000' WHERE waiter_id = 'WA121';
Query OK, 0 rows affected (0.01 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql>
mysql> INSERT INTO TRANSACTION_LOG (TRANSACTION_ID, OPERATION_TYPE, TABLE_NAME, OLD_VALUE, NEW_VALUE)
    -> VALUES ('txn_001', 'UPDATE', 'WAITER', 'SALARY=25000', 'SALARY=30000');
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

LOCK TABLES: This line locks the specified tables (WAITER, EMPLOYEE, TRANSACTION_LOG) for write operations. This means that other processes attempting to write to these tables will be blocked until the tables are unlocked

UPDATE WAITER;: This line updates the SALARY column in the WAITER table, setting the salary to 30000 for the waiter with the ID 'WA121'.

INSERT INTO TRANSACTION_LOG: This line inserts a record into the TRANSACTION_LOG table, recording information about the update operation performed on the WAITER table. It includes details such as the transaction ID, operation type (which is an update in this case), the table name (WAITER), the old value (25000 in the SALARY column), and the new value (30000 in the SALARY column).

UNLOCK TABLES This line unlocks the previously locked tables (WAITER, EMPLOYEE, TRANSACTION_LOG), allowing other processes to access and modify these tables again.

TRANSACTION FAILURE

```
mysql> UPDATE WAITER SET SALARY = '30000' WHERE waiter_id = 'WA121';
Query OK, 0 rows affected (0.01 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql>
mysql> INSERT INTO TRANSACTION_LOG (TRANSACTION_ID, OPERATION_TYPE, TABLE_NAME, OLD_VALUE, NEW_VALUE)
    -> VALUES ('txn_001', 'UPDATE', 'WAITER', 'SALARY=25000', 'SALARY=30000');
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> INSERT INTO TRANSACTION_LOG ( transaction_id, operation_type, table_name, old_value, new_value, timestamp)
    -> VALUES ( 'txn_002', 'UPDATE', 'Waiter', 'Salary=20000', 'Salary=30000', NOW());
Query OK, 1 row affected (0.00 sec)

mysql> UPDATE Waiter SET Salary=30000 WHERE waiter_id = 'WA127';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql>
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

START TRANSACTION;: Begins a new transaction.

INSERT INTO TRANSACTION_LOG: This line attempts to insert a record into the TRANSACTION_LOG table, recording information about an update operation (in this case, increasing the salary from 20000 to 30000 for a waiter). The NOW() function is used to get the current timestamp for the transaction. To Maintain backup data.

UPDATE Waiter SET Salary=30000 WHERE waiter_id = 'WA127 This line tries to update the Salary column in the Waiter table, setting the salary to 30000 for the waiter with ID 'WA127'.

ROLLBACK: This line rolls back (undoes) the changes made during the current transaction. In this context, it means that both the attempted insertion into the TRANSACTION_LOG table and the update to the Waiter table are reverted, as if they never happened.

DEADLOCK PREVENTION

```
Query OK, 0 rows affected (0.00 sec)

mysql> dedlock
    -> SET innodb_lock_wait_timeout = 10;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
 for the right syntax to use near 'dedlock
SET innodb_lock_wait_timeout = 10' at line 1
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM Waiter WHERE waiter_id = 'WA127' FOR UPDATE;
+---------+--------+------------+--------+-----------+
| NAME_ID | GENDER | DOB        | SALARY | WAITER_ID |
+---------+--------+------------+--------+-----------+
| Joshep  | Male   | 24-02-1990 | 30000  | WA127     |
+---------+--------+------------+--------+-----------+
1 row in set (0.00 sec)

mysql> UPDATE waiter SET Salary=30000 WHERE waiter_id = 'WA127';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> INSERT INTO transaction_log (transaction_id, operation_type, table_name, old_value, new_value)
    -> VALUES ( 'txn_002', 'UPDATE', 'Waiter', 'Salary=20000', 'Salary=30000');
Query OK, 1 row affected (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

SET innodb_lock_wait_timeout = 10;: This line sets the timeout period for InnoDB lock waits to 10 seconds. InnoDB is a storage engine for MySQL that provides transaction support.

START TRANSACTION;: Begins a new transaction.

SELECT * FROM Waiter WHERE waiter_id = 'WA127' FOR UPDATE;: This line selects data from the Waiter table for the waiter with ID 'WA127' and locks the selected rows with a "FOR UPDATE" clause. This locking ensures that other transactions cannot modify these rows until the current transaction is completed.

INSERT INTO transaction_log;: This line inserts a record into the transaction_log table, recording information about an update operation (in this case, increasing the salary from 20000 to 30000 for a waiter).

COMMIT;: Commits the transaction, making all changes permanent if all statements within the transaction execute successfully.

Recovery Mechanism:

```
mysql> INSERT INTO transaction_log (transaction_id, operation_type, table_name, old_value, new_value)
    -> VALUES ( 'txn_002', 'UPDATE', 'Waiter', 'Salary=20000', 'Salary=30000');
Query OK, 1 row affected (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> SAVEPOINT checkpoint1;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> UPDATE WAITER SET SALARY = '40000' WHERE NAME_ID = 'WA121';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql>
mysql> INSERT INTO TRANSACTION_LOG (TRANSACTION_ID, OPERATION_TYPE, TABLE_NAME, OLD_VALUE, NEW_VALUE)
    ->
    -> VALUES ('txn_004', 'UPDATE', 'WAITER', 'SALARY=25000', 'SALARY=30000');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> ROLLBACK TO SAVEPOINT checkpoint1;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

RollBack Mechanism in DBMS:

START TRANSACTION; Most DBMSs maintain transaction logs, which record all changes made to the database during transaction execution.

SAVEPOINT checkpoint1 This line creates a savepoint within the transaction. A savepoint is a point in the transaction where you can later roll back to, preserving changes made after the savepoint..

UPDATE WAITER SET SALARY = '40000' WHERE NAME_ID = 'WA121' This line updates the SALARY column in the WAITER table, setting the salary to 40000 for the waiter with the NAME_ID 'WA121'.

INSERT INTO TRANSACTION_LOG This line attempts to insert a record into the TRANSACTION_LOG table, recording information about an update operation (in this case, increasing the salary from 25000 to 30000 for a waiter).

ROLLBACK TO SAVEPOINT checkpoint1; This line rolls back the transaction to the savepoint named checkpoint1. Rolling back to a savepoint undoes changes made after the savepoint while preserving changes made before it. In this case, the update to set the salary to 40000 for waiter 'WA121' is undone..

Related Code Explanation:

In the provided code, the START TRANSACTION; statement initiates a new transaction.

SQL operations are performed within the transaction, such as updates, inserts, or deletes.

The IF statement checks if an error condition occurred during the transaction.

If an error occurred (some_condition evaluates to true), the ROLLBACK SAVEPOINT; statement is executed to rollback the transaction, undoing any changes made by the transaction.

Otherwise, if no error occurred, the COMMIT; statement is executed to commit the transaction, making the changes permanent.

This mechanism ensures that data consistency is maintained even in the face of errors or failures during transaction execution. By using transaction logs and rollback operations, the DBMS can recover the database to a consistent state, ensuring data integrity.

```
Rows matched: 0  Changed: 0  Warnings: 0

mysql>
mysql> INSERT INTO TRANSACTION_LOG (TRANSACTION_ID, OPERATION_TYPE, TABLE_NAME, OLD_VALUE, NEW_VALUE)
    ->
    -> VALUES ('txn_004', 'UPDATE', 'WAITER', 'SALARY=25000', 'SALARY=30000');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> ROLLBACK TO SAVEPOINT checkpoint1;
Query OK, 0 rows affected (0.00 sec)

mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> UPDATE CHEF SET SALARY = SALARY*1.1 WHERE CHEF_ID = 'CH098';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
mysql> INSERT INTO transaction_log (transaction_id, operation_type, table_name, old_value, new_value)
    -> VALUES ('txn_005', 'UPDATE', 'CHEF', 'SALARY=30000', 'SALARY=33000');
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Transaction Log in DBMS:

START TRANSACTION: This line starts a transaction, which means that subsequent SQL statements will be treated as part of a single unit of work. Transactions ensure data consistency by allowing a series of operations to be either committed (saved permanently) or rolled back (reverted) as a whole.

UPDATE CHEF SET SALARY = SALARY*1.1 WHERE CHEF_ID = 'CH098' This line updates the SALARY column in the CHEF table by multiplying the current salary by 1.1 (increasing it by 10%) for the chef with the CHEF_ID 'CH098'.

INSERT INTO transaction_log: This line inserts a record into the transaction_log table, recording information about an update operation (in this case, increasing the salary from 30000 to 33000 for a chef).

Commit: If a transaction completes successfully, the changes made by the transaction are permanently saved to the database when the transaction is committed.

Related Code Explanation:

In the provided code, the START TRANSACTION; statement initiates a new transaction.

SQL operations are performed within the transaction, such as updates, inserts, or deletes.

The IF statement checks if an error condition occurred during the transaction.

If an error occurred, the data from Transaction Log; is used as backup data to find all the transaction made.

CHAPTER-6

CODE FOR THE PROJECT

Home Page:

```
package rest;

import java.awt.EventQueue;
import java.awt.Image;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class MainPage1 {

	JFrame frame;

	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					MainPage1 window = new MainPage1();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}


	public MainPage1() {
		initialize();
	}
	private void initialize() {
		frame = new JFrame();
		frame.setBounds(100, 100, 1024, 540);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	   Image img= new
ImageIcon(this.getClass().getResource("/main1.jpg")).getImage();
		frame.getContentPane().setLayout(null);

			JButton btnNewButton = new JButton("New Employee");
```

```
		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				Employee_add obj = new Employee_add();
				obj.frame2.setVisible(true);
				frame.dispose();
			}
		});
		btnNewButton.setBounds(542, 47, 185, 51);
		frame.getContentPane().add(btnNewButton);

		JButton btnNewButton_1 = new JButton("Billing");
		btnNewButton_1.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				Billing obj = new Billing();
				obj.frame1.setVisible(true);
				frame.dispose();
			}
		});
		btnNewButton_1.setBounds(747, 47, 185, 51);
		frame.getContentPane().add(btnNewButton_1);

		JButton btnNewButton_2 = new JButton("Menu");
		btnNewButton_2.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				Menu obj = new Menu();
				obj.frame1.setVisible(true);
				frame.dispose();
			}
		});
		btnNewButton_2.setBounds(542, 442, 185, 51);
		frame.getContentPane().add(btnNewButton_2);

		JButton btnNewButton_3 = new JButton("Account");
		btnNewButton_3.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				Account obj = new Account();
				obj.frame1.setVisible(true);
				frame.dispose();
			}
		});
		btnNewButton_3.setBounds(747, 442, 185, 51);
		frame.getContentPane().add(btnNewButton_3);

		JLabel label_1 = new JLabel("");
		label_1.setBounds(0, 0, 1024, 512);
		label_1.setIcon(new ImageIcon(img));
		frame.getContentPane().add(label_1);
	}
}
```

Employee_Adding:

```
package rest;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Scanner;

import javax.swing.JTextField;
import javax.swing.JButton;

public class Employee_add {
	String field1,field3,field4;
	int field2;
	JFrame frame2;
	private JTextField textField;
	private JTextField textField_1;
	private JTextField textField_2;
	private JTextField textField_3;
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Employee_add window = new Employee_add();
					window.frame2.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}
	public Employee_add() {
		initialize();
	}


	private void initialize() {
		frame2 = new JFrame();
		frame2.setBounds(100, 100, 450, 300);
		frame2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame2.getContentPane().setLayout(null);

		JLabel lblNewLabel = new JLabel("Gender");
```

```
lblNewLabel.setFont(new Font("Tahoma", Font.PLAIN, 15));
lblNewLabel.setBounds(30, 29, 98, 38);
frame2.getContentPane().add(lblNewLabel);

JLabel lblSalary = new JLabel("Salary");
lblSalary.setFont(new Font("Tahoma", Font.PLAIN, 15));
lblSalary.setBounds(30, 77, 98, 38);
frame2.getContentPane().add(lblSalary);

JLabel lblName = new JLabel("Name");
lblName.setFont(new Font("Tahoma", Font.PLAIN, 15));
lblName.setBounds(30, 125, 98, 38);
frame2.getContentPane().add(lblName);

JLabel lblEmployeeid = new JLabel("Employee_ID");
lblEmployeeid.setFont(new Font("Tahoma", Font.PLAIN, 14));
lblEmployeeid.setBounds(30, 173, 98, 38);
frame2.getContentPane().add(lblEmployeeid);

textField = new JTextField();
textField.setBounds(160, 29, 145, 29);
frame2.getContentPane().add(textField);
textField.setColumns(10);

textField_1 = new JTextField();
textField_1.setColumns(10);
textField_1.setBounds(160, 77, 145, 29);
frame2.getContentPane().add(textField_1);

textField_2 = new JTextField();
textField_2.setColumns(10);
textField_2.setBounds(160, 125, 145, 29);
frame2.getContentPane().add(textField_2);

textField_3 = new JTextField();
textField_3.setColumns(10);
textField_3.setBounds(160, 180, 145, 29);
frame2.getContentPane().add(textField_3);

JButton btnNewButton = new JButton("Next");
btnNewButton.setFont(new Font("Tahoma", Font.PLAIN, 15));
btnNewButton.setBounds(297, 219, 109, 34);
btnNewButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                try {
                        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root",
"teamosekire");
                        Scanner scan=new Scanner(System.in);
                        field1=textField.getText();
```

```
                                        field2=Integer.parseInt(textField_1.getText());
                                        field3=textField_2.getText();
                                        field4=textField_3.getText();
                                        String mini="insert into Employee values(?,?,?,?)";
                                        PreparedStatement s3= con.prepareStatement(mini);
                                        s3.setString(1,field1);
                                        s3.setInt(2, field2);
                                        s3.setString(3, field3);
                                        s3.setString(4, field4);
                                        s3.executeUpdate();
                                        s3.close();
                                        con.close();
                                        Employee_D obj = new Employee_D();
                                        obj.frame1.setVisible(true);
                                        frame2.dispose();
                                } catch (Exception e1) {
                                        System.out.println(e1.getMessage());
                                }

                        }
                });
                frame2.getContentPane().add(btnNewButton);
        }
}
```

Employee Display:

```
package rest;

import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;


public class Employee_D {

        JFrame frame1;
        private JTable jtbl;
        private DefaultTableModel model;
```

```
/**
 * Launch the application.
 */
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
		public void run() {
			try {
				Employee_D window = new Employee_D();
				window.frame1.setVisible(true);
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	});
}

/**
 * Create the application.
 */
public Employee_D() {
	initialize();
}

/**
 * Initialize the contents of the frame.
 */
private void initialize() {
	frame1 = new JFrame();
	frame1.setBounds(100, 100, 450, 300);
	frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

	model = new DefaultTableModel();
	model.addColumn("Name");
	model.addColumn("Gender");
	model.addColumn("Salary");
	model.addColumn("Employee_id");

	jtbl = new JTable(model);
	frame1.getContentPane().setLayout(new BorderLayout());
	frame1.getContentPane().add(new JScrollPane(jtbl), BorderLayout.CENTER);

	JButton btnLoadData = new JButton("Back");
	btnLoadData.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
			MainPage1 obj = new MainPage1();
			obj.frame.setVisible(true);
			frame1.dispose();
		}
	});
```

```
		frame1.getContentPane().add(btnLoadData, BorderLayout.SOUTH);
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root",
"teamosekire");
			String query = "SELECT * FROM employee";
			PreparedStatement s1 = con.prepareStatement(query);
			ResultSet rs = s1.executeQuery();
			while (rs.next()) {
				model.addRow(new Object[]{rs.getString("Name"),
rs.getString("Gender"), rs.getString("Salary"), rs.getString("Employee_id")});
			}
			con.close();
			s1.close();
			rs.close();
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}
		frame1.pack();
	}




}
```

Menu Display:

```
package rest;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

public class Menu {

	JFrame frame1;
```

```
	private JTable jtbl;
	private DefaultTableModel model;


	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Menu window = new Menu();
					window.frame1.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}


	public Menu() {
		initialize();
	}
	private void initialize() {
		frame1 = new JFrame();
		frame1.setBounds(100, 100, 450, 300);
		frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		model = new DefaultTableModel();
		model.addColumn("Timing");
		model.addColumn("Cuisine");
		model.addColumn("Price");
		model.addColumn("Food_ID");

		jtbl = new JTable(model);
		frame1.getContentPane().setLayout(new BorderLayout());
		frame1.getContentPane().add(new JScrollPane(jtbl), BorderLayout.CENTER);

		JButton btnBack = new JButton("Back");
		btnBack.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				MainPage1 obj = new MainPage1();
				obj.frame.setVisible(true);
				frame1.dispose();
			}
		});
		frame1.getContentPane().add(btnBack, BorderLayout.SOUTH);
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root",
"teamosekire");
```

```
                        String query = "SELECT * FROM Menu";
                        PreparedStatement s1 = con.prepareStatement(query);
                        ResultSet rs = s1.executeQuery();
                        while (rs.next()) {
                                model.addRow(new Object[]{rs.getString("Timing"),
rs.getString("Cuisine"), rs.getString("Price"), rs.getString("Food_ID")});
                        }
                        con.close();
                        s1.close();
                        rs.close();
                } catch (Exception e) {
                        System.out.println(e.getMessage());
                }


                frame1.pack();
        }




}
```

Billing:

```
package rest;

import java.awt.EventQueue;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.time.Instant;
import java.util.Scanner;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JButton;

public class Billing {
    int field1,field2,field3,field4,field5,field6,field7;
    int n,id,Quantity;
    String cost,service,tax;
        JFrame frame1;
        private JTextField textField;
        private JTextField textField_1;
```

```
 private JTextField textField_2;
 private JTextField textField_3;
 private JTextField textField_4;
 private JTextField textField_5;
 private JTextField textField_6;
 private JTextField textField_7;
 private JTextField textField_8;
 private JTextField textField_9;
 private JTextField textField_10;
 private JTextField textField_11;
 private JTextField textField_12;
 private JButton btnNewButton_1;
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
		public void run() {
			try {
				Billing window = new Billing();
				window.frame1.setVisible(true);
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	});
}


public Billing() {
	initialize();
}


private void initialize() {
	frame1 = new JFrame();
	frame1.setBounds(100, 100, 1024, 540);
	frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   Image img= new ImageIcon(this.getClass().getResource("/Bill.jpg")).getImage();
   frame1.getContentPane().setLayout(null);

	textField = new JTextField();
	textField.setBounds(214, 113, 113, 34);
	frame1.getContentPane().add(textField);
	textField.setColumns(10);

	textField_1 = new JTextField();
	textField_1.setColumns(10);
	textField_1.setBounds(214, 160, 113, 34);
	frame1.getContentPane().add(textField_1);

	textField_2 = new JTextField();
	textField_2.setColumns(10);
```

```
textField_2.setBounds(214, 209, 113, 34);
frame1.getContentPane().add(textField_2);

textField_3 = new JTextField();
textField_3.setColumns(10);
textField_3.setBounds(214, 257, 113, 34);
frame1.getContentPane().add(textField_3);

textField_4 = new JTextField();
textField_4.setColumns(10);
textField_4.setBounds(214, 353, 113, 34);
frame1.getContentPane().add(textField_4);

textField_5 = new JTextField();
textField_5.setColumns(10);
textField_5.setBounds(214, 401, 113, 34);
frame1.getContentPane().add(textField_5);

textField_6 = new JTextField();
textField_6.setColumns(10);
textField_6.setBounds(214, 305, 113, 34);
frame1.getContentPane().add(textField_6);

textField_7 = new JTextField();
textField_7.setBounds(773, 115, 210, 40);
frame1.getContentPane().add(textField_7);
textField_7.setColumns(10);

textField_8 = new JTextField();
textField_8.setColumns(10);
textField_8.setBounds(773, 169, 210, 40);
frame1.getContentPane().add(textField_8);

textField_9 = new JTextField();
textField_9.setColumns(10);
textField_9.setBounds(773, 261, 210, 40);
frame1.getContentPane().add(textField_9);

textField_10 = new JTextField();
textField_10.setColumns(10);
textField_10.setBounds(773, 305, 210, 40);
frame1.getContentPane().add(textField_10);

textField_11 = new JTextField();
textField_11.setColumns(10);
textField_11.setBounds(773, 351, 210, 40);
frame1.getContentPane().add(textField_11);

textField_12 = new JTextField();
textField_12.setColumns(10);
```

```
		textField_12.setBounds(773, 216, 210, 40);
		frame1.getContentPane().add(textField_12);

		JButton btnNewButton = new JButton("Calculate");
		btnNewButton.setBounds(8, 447, 174, 46);
		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				try {
					Class.forName("com.mysql.cj.jdbc.Driver");
					Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root",
"teamosekire");
					Scanner scan=new Scanner(System.in);
					field1=Integer.parseInt(textField.getText());
					field2=Integer.parseInt(textField_1.getText());
					field3=Integer.parseInt(textField_2.getText());
					field4=Integer.parseInt(textField_3.getText());
					field5=Integer.parseInt(textField_6.getText());
					field6=Integer.parseInt(textField_4.getText());
					field7=Integer.parseInt(textField_5.getText());
					 n = (field1 * 30) + (field2 * 45) + (field3 * 55) +
(field4 * 20) + (field5 * 50) + (field6 * 15) + (field7 * 20);

Quantity=field1+field2+field3+field4+field5+field6+field7;
					 cost="BL";
					 id=456;
					 id++;
					 service = cost + String.valueOf(id);
					Instant t1=java.time.Clock.systemUTC().instant();
					String time = t1.toString();
					textField_7.setText(service);
					textField_8.setText(time);
					textField_12.setText(String.valueOf(n));
					textField_9.setText(String.valueOf(100));
					textField_10.setText(String.valueOf(n*0.12));
					textField_11.setText(String.valueOf(n+(n*0.12)+100));
					String mini="insert into Billing values(?,?,?,?)";
					PreparedStatement s3= con.prepareStatement(mini);
					s3.setString(1,service);
					s3.setInt(2, Quantity);
					s3.setString(3, time);
					s3.setInt(4, (int) (n+(n*0.12)+100));
					s3.executeUpdate();
					s3.close();

					scan.close();
					con.close();

				} catch (Exception e1) {
					System.out.println(e1.getMessage());
```

```
                }
            }
        });
        frame1.getContentPane().add(btnNewButton);

        JButton btnNewButton_2 = new JButton("Clear");
        btnNewButton_2.setBounds(202, 447, 133, 46);
        btnNewButton_2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                textField.setText("");
                textField_1.setText("");
                textField_2.setText("");
                textField_3.setText("");
                textField_4.setText("");
                textField_5.setText("");
                textField_6.setText("");
                textField_7.setText("");
                textField_8.setText("");
                textField_9.setText("");
                textField_10.setText("");
                textField_11.setText("");
                textField_12.setText("");
            }
        });
        frame1.getContentPane().add(btnNewButton_2);

        JButton btnNewButton_2_1 = new JButton("Back");
        btnNewButton_2_1.setBounds(354, 447, 127, 46);
        btnNewButton_2_1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                MainPage1 obj = new MainPage1();
                obj.frame.setVisible(true);
                frame1.dispose();
            }
        });
        frame1.getContentPane().add(btnNewButton_2_1);
        JLabel label_1 = new JLabel("");
        label_1.setBounds(0, 0, 1010, 503);
        label_1.setIcon(new ImageIcon(img));
        frame1.getContentPane().add(label_1);        }
}
```

Account Display:

```
package rest;

import java.awt.BorderLayout;
import java.awt.EventQueue;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

public class Account {

	 JFrame frame1;
	 private JTable jtbl;
	 private DefaultTableModel model;
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Account window = new Account();
					window.frame1.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}


	public Account() {
		initialize();
	}


	private void initialize() {
		frame1 = new JFrame();
		frame1.setBounds(100, 100, 450, 300);
		frame1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		model = new DefaultTableModel();
		model.addColumn("Bill_ID");
		model.addColumn("Total_count");
		model.addColumn("Handled_By");
		model.addColumn("Table_ID");

		jtbl = new JTable(model);
		frame1.getContentPane().setLayout(new BorderLayout());
		frame1.getContentPane().add(new JScrollPane(jtbl), BorderLayout.CENTER);
```

```
		JButton btnBack = new JButton("Back");
		btnBack.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				MainPage1 obj = new MainPage1();
				obj.frame.setVisible(true);
				frame1.dispose();
			}
		});
		frame1.getContentPane().add(btnBack, BorderLayout.SOUTH);
		try {
			Class.forName("com.mysql.cj.jdbc.Driver");
			Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root",
"teamosekire");
			String query = "SELECT * FROM BILL";
			PreparedStatement s1 = con.prepareStatement(query);
			ResultSet rs = s1.executeQuery();
			while (rs.next()) {
				model.addRow(new Object[]{rs.getString("Bill_ID"),
rs.getString("Total_count"), rs.getString("Handled_By"), rs.getString("Table_ID")});
			}
			con.close();
			s1.close();
			rs.close();
		} catch (Exception e) {
			System.out.println(e.getMessage());
		}


		frame1.pack();
	}

}
```

CHAPTER-7

RESULT AND DISCUSSION

The Restaurant Management System is an essential component of efficient restaurant operations, ensuring smooth customer service and streamlined processes. This system encompasses various modules to handle customer data, menu management, order processing, billing calculations, and payment processing.

ADVERTISMENTS Table: This table stores information about advertisements, including the deadline, advertisement area, theme, budget, and customer ID.

BILL Table: This table tracks bills with a unique bill ID, total count, the person who handled the bill, and a table ID.In the billing system, this table could be used to store information about generated bills, their handling, and associated details.

CASHIER Table: The cashier table contains data about cashiers, including their admin ID, associated bill ID, food items, customer name, and price.In the billing system, this table might be used to track cashier activities related to bill payments or other transactions.

CHEF Table: This table holds information about chefs, including their gender, date of birth, name, salary, and chef ID.

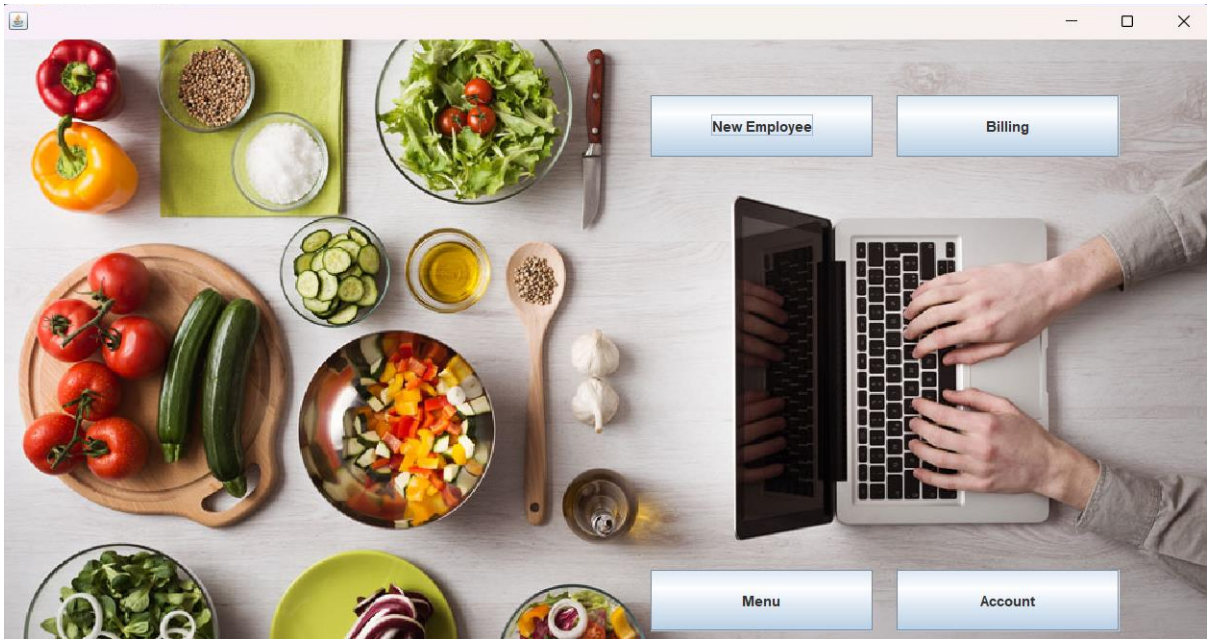
FEEDBACK Table: This table records feedback from customers regarding food quality, service, favorite food, improvement suggestions, and customer ID.In the billing system context, feedback mechanisms might be used for customer satisfaction surveys or service improvement initiatives..

FOOD_DELIVERY Table: This table contains information about food deliveries, including customer ID, total price, items details, delivery partner, and branch ID.

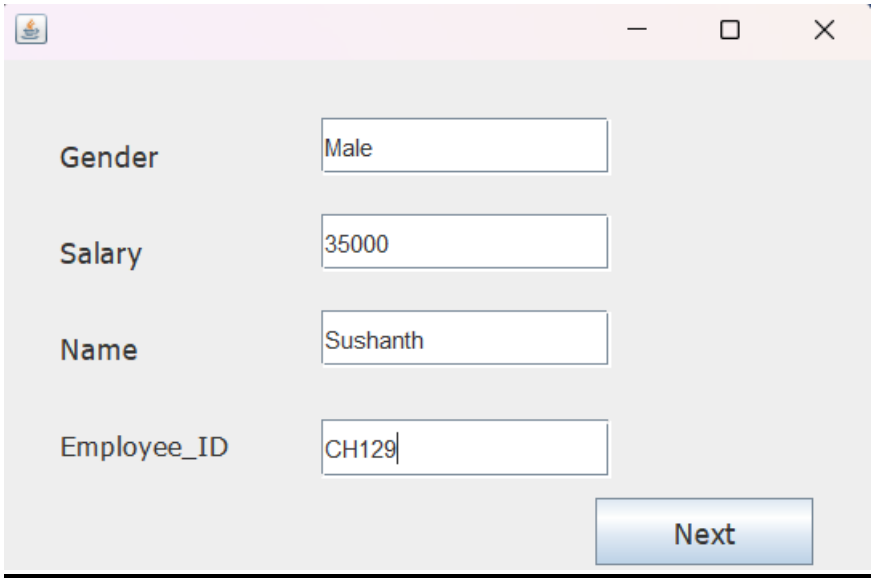
Database Management: The system maintains a robust database architecture for data storage, retrieval, and backup. It ensures data integrity, security, and scalability to handle large volumes of customer information and transactional data effectively.

In summary, the Restaurant Management System streamlines restaurant operations, enhances customer satisfaction, and supports data-driven decision-making through efficient database management and reporting capabilities.

Main Page:-



New Employee:-



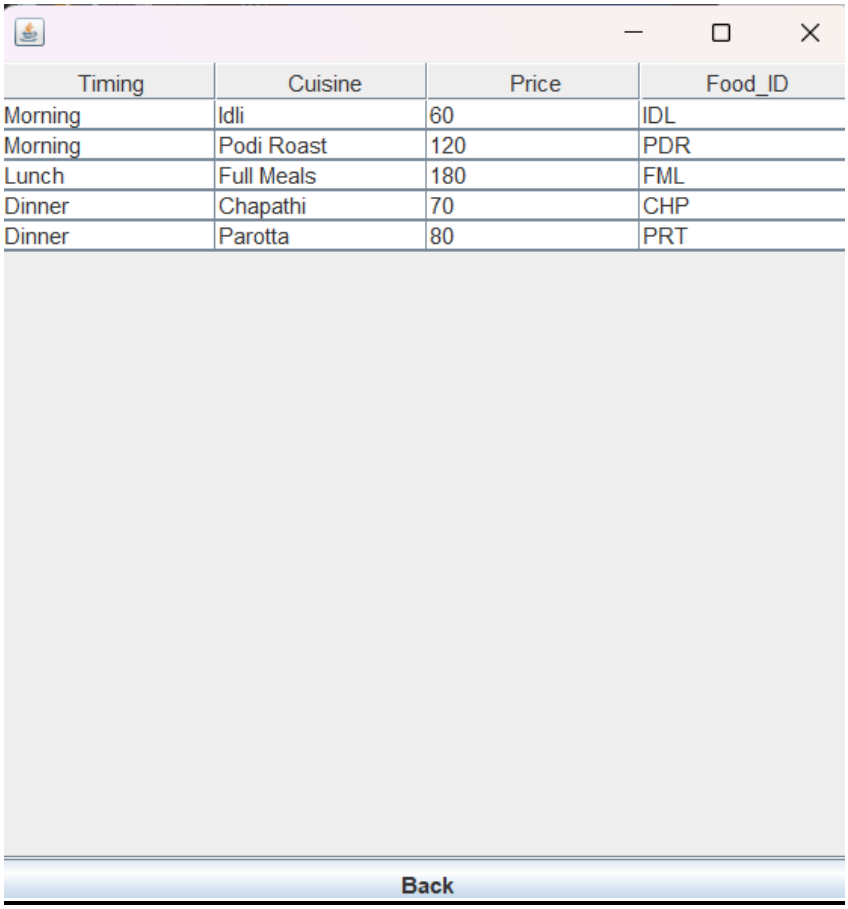
Employee Display: -

Name	Gender	Salary	Employee_id
Ram	Male	25000	WA121
Joshep	Male	20000	WA127
Karupusamy	Male	20000	WA098
Mythli	Female	20000	WA112
Jeniferg	Female	20000	WA089
Arul	Male	40000	CH078
Arunachalam	Male	30000	CH118
Mahendran	Male	30000	CH056
keerthana	Female	400000	CH098
Anbarasan	Male	60000	CH110
Rishi	Male	80000	MG081
Madan	Male	80000	MG099
Shilpa	Female	90000	MG104
Sherma	Female	80000	MG112
Ritul	Female	70000	MG088
Pranav	Male	30000	CH128
Sanjay	Male	25000	WA143
Pranav	Male	30000	WA123
Sushanth	Male	35000	CH129

Back

Billing:-

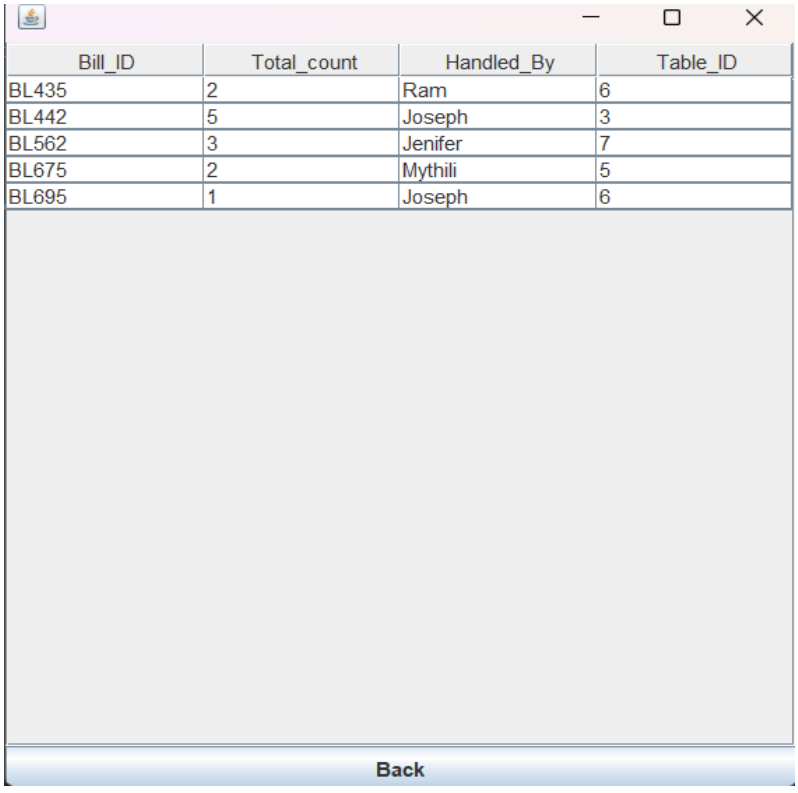
Menu:-



Timing	Cuisine	Price	Food_ID
Morning	Idli	60	IDL
Morning	Podi Roast	120	PDR
Lunch	Full Meals	180	FML
Dinner	Chapathi	70	CHP
Dinner	Parotta	80	PRT

Back

Account Display:-



Bill_ID	Total_count	Handled_By	Table_ID
BL435	2	Ram	6
BL442	5	Joseph	3
BL562	3	Jenifer	7
BL675	2	Mythili	5
BL695	1	Joseph	6

Back

FUTURE SCOPE AND LIMITATIONS

Software developers may not expect. The following principles enhances extensibility like hide data structure, avoid traversing multiple.

Links or methods avoid case statements on object type and distinguish public and private operations.

• Reusability: Reusability is possible as and when require in this application. We can update it next version. Reusable software reduces design, coding and testing cost by amortizing effort

Over several designs. Reducing the amount of code also simplifies understanding, which increases the likelihood that the code is correct. We follow up both types of reusability:

Sharing of newly written code within a project and reuse of previously written code on new projects.

• Understand ability: A method is understandable if someone other than the creator of the method can understand the code (as well as the creator after a time lapse). We use the method, which small and coherent helps to accomplish this.

• Cost-effectiveness: Its cost is under the budget and make within given time period. It is desirable to aim for a system with a minimum cost subject to the condition that it must satisfy the entire requirement. Scope of this document is to put down the requirements, clearly identifying the information needed by the user, the source of the information and outputs expected from the system.

LIMITATIONS:-

- This application cannot be accessed remotely.
- This application requires knowledgeable person to use this application.
- This application does not have journals

CONCLUSION

After all the hard work is done for the Restaurant Management System, it is a software designed to streamline restaurant operations, including managing billing cycles, paying bills, and overseeing various operational details. This software significantly reduces the need for manual data entry, leading to greater efficiency in restaurant management. Its user-friendly interface makes it easy for anyone to use, thereby decreasing the time required for administrative tasks and enhancing overall productivity.

If already registered, click to check your payment status

Course outline

- About NPTEL
- Introduction to Database Systems
- Week 1
- Week 2
- Week 3
- Week 4
- Week 5
- Week 6
- Week 7
- Week 8
- Week 9
- Week 10
- Week 11
- Week 12

Date enrolled 2024-01-29

Email pranavt1805@gmail.com

Name Pranav T

11.21%

Course Progress

Unit wise Progress

Assessment scores

Assignment 1:	**98.0**
Assignment 2:	**100.0**
Assignment 3:	**100.0**
Week 4 : Assignment 4:	**100.0**
Week 5 : Assignment 5:	**70.0**
Week 6 : Assignment 6:	**60.0**
Week 7 : Assignment 7:	**60.0**
Week 8 : Assignment 8:	**40.0**

Announcement:
You are currently receiving course related emails. Click here to unsubscribe.

Discussion forum:
If you want to unsubscribe from forum Click here

If already registered, click to check your payment status

Course outline

- About NPTEL
- Introduction to Database Systems
- Week 1
- Week 2
- Week 3
- Week 4
- Week 5
- Week 6
- Week 7
- Week 8
- Week 9
- Week 10
- Week 11
- Week 12

Date enrolled 2024-01-29

Email pranavt1805@gmail.com

Name Pranav T

11.21%

Course Progress

Unit wise Progress

Assessment scores

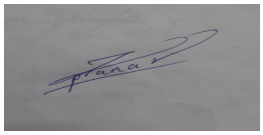
Week 5 : Assignment 5:	**70.0**
Week 6 : Assignment 6:	**60.0**
Week 7 : Assignment 7:	**60.0**
Week 8 : Assignment 8:	**40.0**
Week 9 : Assignment 9:	**80.0**
Week 10 : Assignment 10:	**50.0**
Week 11 : Assignment 11:	**80.0**
Week 12 : Assignment 12:	**20.0**

Announcement:
You are currently receiving course related emails. Click here to unsubscribe.

Discussion forum:
If you want to unsubscribe from forum Click here

MORNING SESSION (FN)	National Programme on Technology Enhanced Learning
Hall Ticket For	2024 Apr: CS55 Introduction to Database Systems - Online

Candidate Name	Pranav T				
Roll No	NOC24CS55S553400299		Seating Number	53400299	
Date of Birth	18-05-2005				
PwD Status	No	Compensatory Time Required	N.A	Scribe Required	N.A
Exam Date	Sunday, 21 April, 2024				
Reporting Time	08:00 am		Gate Closure	09:30 am	
Exam Timing	09:00 am		Shift	FN	
Test Centre Name	iON Digital Zone iDZ Kovilambakkam				
Test Centre Address	Fortune Towers,1st and 2nd Floor,SH 109, 200 Feet Thoraipakkam Pallavaram Radial Road, Near Eachangadu signal,Kovilambakkam, Chennai, Tamil Nadu, India - 600117				



NPTEL Coordinator

NPTEL EXAM - 21 APRIL, 2024
General instructions for candidates - FN
(All timings mentioned here are in IST)

The total duration of the examination is 180 minutes.
Candidates will be permitted to leave the examination hall only after 10:30 am, on a need basis.

Hall ticket and Entry:

1. The Hall Ticket must be presented for verification along with one original photo identification (not photocopy or scanned copy). Examples of acceptable photo identification documents are School ID, College ID, Employee ID, Driving License, Passport, PAN card, Voter ID, and Aadhaar-ID. A printed copy of the hall ticket and original photo ID card should be brought to the exam centre. Hall ticket and ID card copies on the phone will not be permitted.

2. This Hall Ticket is valid only if the candidate's photograph and signature images are legible. To ensure this, print the Hall Ticket on A4-sized paper using a laser printer, preferably a color photo printer.

3. **TIMELINE:** 8:00 am - Report to the examination venue | 8:40 am – Candidates will be permitted to occupy their allotted seats| 8:50 am – Candidates can login and start reading instructions prior to the examination | 9:00 am - Exam starts | 9:30 am - Gate closes, candidates will not be allowed after this time | 10:30 am Submit button will be enabled | 12:00 pm exam ends.

4. Candidates will be permitted to appear for the examination ONLY after their credentials are verified by center officials.

P.T.O.

5. Candidates are advised to locate the examination center at least a day prior to the examination, so that they can reach the center on time for the examination.

STATIONERY REQUIREMENTS:

- A4 sheets will be provided to candidates for rough work. Candidates have to write their name and registration number on the A4 Sheets before they start using it. The A4 sheets must be returned to the invigilator at the end of the examination.

- On-screen calculator will be available during the exam. Candidates are advised to familiarize themselves with this virtual Scientific calculator well ahead of the exam.
  Link: https://www.tcsion.com/OnlineAssessment/ScientificCalculator/Calculator.html

- You should bring your own pen/pencil; it will NOT be given at the examination centre.

DRESS CODE:

- Candidates are expected to come in professional attire to write the exams.

- Candidates wearing SHORTS will NOT be permitted inside the exam hall.

PERMITTED:

- You may bring vehicle keys inside the exam hall.

- You are advised to carry your own drinking water in a transparent bottle.

- Candidates are allowed to bring sanitizer in a small transparent bottle.

NOT PERMITTED:

- Watches, wallets, mobile phones, Bluetooth devices, microphones, pagers, health bands or any other electronic gadgets, any printed/blank/handwritten paper, log tables, writing pads, scales, geometry/pencil-boxes, pouches, calculators, pen drives, electronic pens, handbags, goggles, electronic vehicle keys or similar such items are NOT allowed inside the examination centre. There may not be any facility for the safekeeping of these devices outside the examination hall; it will be prudent not to bring valuables to the examination center. Candidates will not be permitted to carry any food items in the exam centre. We suggest that you bring a bag to keep routine belongings outside the exam hall. Neither NPTEL nor the exam provider takes responsibility for the bag and the belongings. You may keep it outside at your own risk.

MANDATORY :

- Hall tickets have to be returned to the invigilator before leaving the exam hall. No paper can be taken out of the exam hall.

- Press the SUBMIT button on the computer after you have completed the exam.

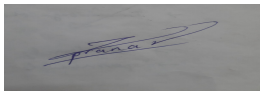
IMPORTANT:

- A basic code of conduct during the exam should be followed, failing which, NPTEL reserves the right to take appropriate action.

- In case the exam is delayed due to any unforeseen circumstances, NPTEL will decide on the appropriate course of action as it deems fit.

AT THE EXAM CENTRE, IF YOU ENCOUNTER ANY ISSUES WITH RESPECT TO THE COMPUTER OR EXAM OFFICIALS, KINDLY CONTACT THE NPTEL EXAM REPRESENTATIVE, WHO WILL BE AVAILABLE AT THE CENTRE.

I HEREBY ACKNOWLEDGE THAT I HAVE READ, UNDERSTOOD AND AGREE TO FOLLOW THE ABOVE MENTIONED INSTRUCTIONS.



Signature of the Candidate

If already registered, click to check your payment status

Course outline

- About NPTEL
- Introduction to Database Systems
- Week 1
- Week 2
- Week 3
- Week 4
- Week 5
- Week 6
- Week 7
- Week 8
- Week 9
- Week 10
- Week 11

Date enrolled 2024-01-29

Email rk0992@srmist.edu.in

Name Rashmiya km

9.35%

Course Progress

Unit wise Progress

Assessment scores

Week 5 : Assignment 5:	**70.0**
Week 6 : Assignment 6:	**60.0**
Week 7 : Assignment 7:	**60.0**
Week 8 : Assignment 8:	**40.0**
Week 9 : Assignment 9:	-
Week 10 : Assignment 10:	-
Week 11 : Assignment 11:	-
Week 12 : Assignment 12:	**20.0**

Announcement:
You are currently receiving course related emails. Click here to unsubscribe.

If already registered, click to check your payment status

Course outline

- About NPTEL
- Introduction to Database Systems
- Week 1
- Week 2
- Week 3
- Week 4
- Week 5
- Week 6
- Week 7
- Week 8
- Week 9
- Week 10
- Week 11

Date enrolled 2024-01-29

Email rk0992@srmist.edu.in

Name Rashmiya km

9.35%

Course Progress

Unit wise Progress

Assessment scores

Assignment 1:	**98.0**
Assignment 2:	**100.0**
Assignment 3:	**100.0**
Week 4 : Assignment 4:	**100.0**
Week 5 : Assignment 5:	**70.0**
Week 6 : Assignment 6:	**60.0**
Week 7 : Assignment 7:	**60.0**
Week 8 : Assignment 8:	**40.0**

Announcement:
You are currently receiving course related emails. Click here to unsubscribe.

Discussion forum:

AFTERNOON SESSION (AN)	National Programme on Technology Enhanced Learning	NPTEL
Hall Ticket For	2024 Apr: CS55 Introduction to Database Systems - Online	

Candidate Name	Rashmiya km				
Roll No	NOC24CS55S653405019		Seating Number	53405019	
Date of Birth	30-03-2005				
PwD Status	No	Compensatory Time Required	N.A	Scribe Required	N.A
Exam Date	Sunday, 21 April, 2024				
Reporting Time	01:00 pm		Gate Closure	02:30 pm	
Exam Timing	02:00 pm		Shift	AN	
Test Centre Name	Sri Sai Ram Engineering College				
Test Centre Address	Sai Leo Nagar, Sairam Rd, West Tambaram, , Chennai, Tamil Nadu, India - 600044				

NPTEL Coordinator

NPTEL EXAM - 21 APRIL, 2024
General instructions for candidates - AN
(All timings mentioned here are in IST)

The total duration of the examination is 180 minutes.
Candidates will be permitted to leave the examination hall only after 03:30 pm, on a need basis.

Hall ticket and Entry:

1. The Hall Ticket must be presented for verification along with one original photo identification (not photocopy or scanned copy). Examples of acceptable photo identification documents are School ID, College ID, Employee ID, Driving License, Passport, PAN card, Voter ID, and Aadhaar-ID. A printed copy of the hall ticket and original photo ID card should be brought to the exam centre. Hall ticket and ID card copies on the phone will not be permitted.

2. This Hall Ticket is valid only if the candidate's photograph and signature images are legible. To ensure this, print the Hall Ticket on A4-sized paper using a laser printer, preferably a color photo printer.

3. **TIMELINE:** 1:00 pm - Report to the examination venue | 1:40 pm – Candidates will be permitted to occupy their allotted seats| 1:50 pm – Candidates can login and start reading instructions prior to the examination | 2:00 pm - Exam starts | 2:30 pm - Gate closes, candidates will not be allowed after this time | 3:30 pm Submit button will be enabled | 5:00 pm exam ends.

4. Candidates will be permitted to appear for the examination ONLY after their credentials are verified by center officials.

P.T.O.

5. Candidates are advised to locate the examination center at least a day prior to the examination, so that they can reach the center on time for the examination.

STATIONERY REQUIREMENTS:

- A4 sheets will be provided to candidates for rough work. Candidates have to write their name and registration number on the A4 Sheets before they start using it. The A4 sheets must be returned to the invigilator at the end of the examination.

- On-screen calculator will be available during the exam. Candidates are advised to familiarize themselves with this virtual Scientific calculator well ahead of the exam.
  Link: https://www.tcsion.com/OnlineAssessment/ScientificCalculator/Calculator.html

- You should bring your own pen/pencil; it will NOT be given at the examination centre.

DRESS CODE:

- Candidates are expected to come in professional attire to write the exams.

- Candidates wearing SHORTS will NOT be permitted inside the exam hall.

PERMITTED:

- You may bring vehicle keys inside the exam hall.

- You are advised to carry your own drinking water in a transparent bottle.

- Candidates are allowed to bring sanitizer in a small transparent bottle.

NOT PERMITTED:

- Watches, wallets, mobile phones, Bluetooth devices, microphones, pagers, health bands or any other electronic gadgets, any printed/blank/handwritten paper, log tables, writing pads, scales, geometry/pencil-boxes, pouches, calculators, pen drives, electronic pens, handbags, goggles, electronic vehicle keys or similar such items are NOT allowed inside the examination centre. There may not be any facility for the safekeeping of these devices outside the examination hall; it will be prudent not to bring valuables to the examination center. Candidates will not be permitted to carry any food items in the exam centre. We suggest that you bring a bag to keep routine belongings outside the exam hall. Neither NPTEL nor the exam provider takes responsibility for the bag and the belongings. You may keep it outside at your own risk.

MANDATORY :

- Hall tickets have to be returned to the invigilator before leaving the exam hall. No paper can be taken out of the exam hall.

- Press the SUBMIT button on the computer after you have completed the exam.

IMPORTANT:

- A basic code of conduct during the exam should be followed, failing which, NPTEL reserves the right to take appropriate action.

- In case the exam is delayed due to any unforeseen circumstances, NPTEL will decide on the appropriate course of action as it deems fit.

AT THE EXAM CENTRE, IF YOU ENCOUNTER ANY ISSUES WITH RESPECT TO THE COMPUTER OR EXAM OFFICIALS, KINDLY CONTACT THE NPTEL EXAM REPRESENTATIVE, WHO WILL BE AVAILABLE AT THE CENTRE.

I HEREBY ACKNOWLEDGE THAT I HAVE READ, UNDERSTOOD AND AGREE TO FOLLOW THE ABOVE MENTIONED INSTRUCTIONS.

Signature of the Candidate