GROUP NAME: Heisenberg

CLIENT NAME: Los Pollos Hermanos [Synthetic Client]

TEAM MEMBERS: [sorted alphabetically by last name]:

- Adikkesavelu, Aasvitha
- Gade, Rakshit
- Padimiti, Phanidhiraj
- Ramasubbu, Harini
- Thondepu, Manish Kumar
- Vanguru, Karthikeya Reddy

TABLE OF CONTENTS:

Chapter-1 :Requirement Analysis---------- 3

Chapter-2: Conceptual Schema----------5

Chapter-3:Relational Schema ----------13

Chapter-4:SQL Queries ----------27

Chapter-5:Triggers and Procedures ----------72

Chapter-6: User Interface ---------- 78

Chapter-7: Implementation plan----------87

Appendix A ----------89

Reference ----------89

Chapter 1:

Business Requirements:

Client: Los Pollos Hermanos

Requirement Analysis:

Industry	Fast food restaurants (chain)
User	Branch manager
Platform	Web
User goal	To have a secured platform to view and manage operations, suppliers, performance, payments, and employees of a branch
Performance metrics	- Number of orders placed - Number of menu items managed or added - Number of promotions managed or added - Number of reservations managed or added - Number of time employees managed or added - Number of salary analysis requests - Number of best customer analysis requests - Number of delivery summary analysis requests (as per specified timeframe)

Requirements:

To have a platform for a branch manager to perform Operational Management, Performance Analysis, Financial Oversight, Employee Management, and Supplier Management

When a customer places an order, we collect the name(first and last), phone number, emailID, and a customerID which is allocated in a sequence order. The order details are stored along with the date and amount; additionally an ID is assigned to the order. Along with the order details, the payment details such as payment amount, status of payment, date of payment, and payment method are stored. We also must record the feedback that a customer can give for each order they place with the details such as the rating that they provide, comments, and date. For each order, a customer can give multiple feedback.

A customer can be a part of a loyalty program which is not mandatory, and every customer can be enrolled to a single loyalty program at a single time. For each membership, the membership ID, start date, end date, and description are stored. Customers can also make reservations which store the details such as the reservation ID, party size, contact number, reservation time, and preferences.

There are multiple branches of the food chain and hence details of each of the branches or location such as the phone numbers, branch name, and address are stored. Each branch/location is allocated with a unique ID.

Each event can be hosted at the locations and for each event the event ID, event name, date, and description are stored. Each location is also mapped to a warehouse and the details of the warehouse including the warehouse ID, address, capacity, and phone number of the warehouse are stored.

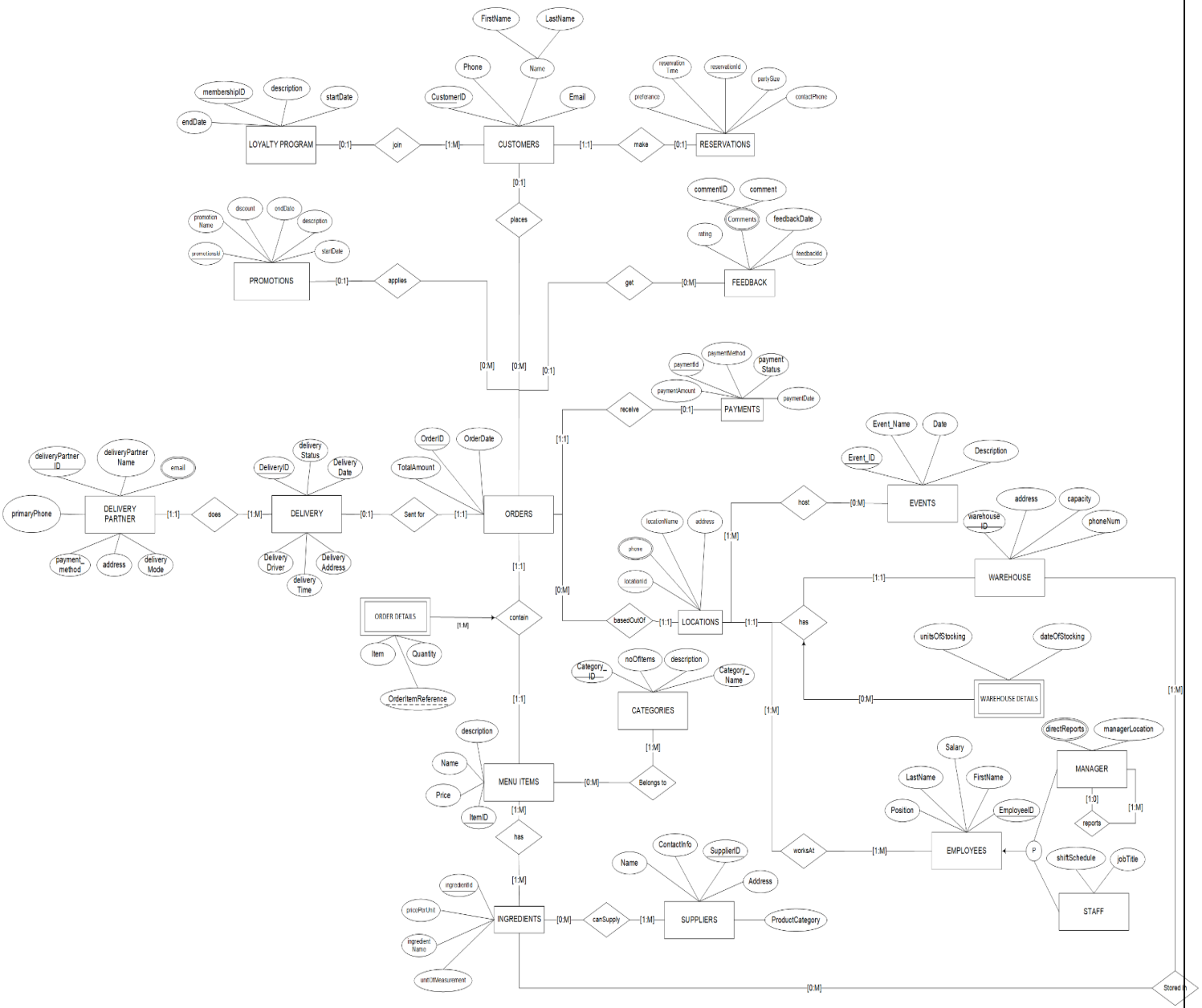
Some of the orders are placed online for which delivery details such as the delivery personal name, delivery address, date, status, and time of delivery are stored and these are attributed to each delivery with a unique ID. To deliver the orders there are certain delivery partners who facilitate the delivery and the details of the delivery partners are stored which includes the primary phone number, payment method, address of the delivery partner, delivery mode, email address which can be multiple, and the delivery partner name.

The food chain has a common menu and the details of the menu items are stored including the name of the menu item, price, description, and each of the menu items is attributed to a unique ID. Menu items are made of ingredients and details such as ingredient ID, name, price per unit, and unit of measurement for each ingredient is stored. Each order contains a list of menu items including the name of the menu item and quantity. Each menu item belongs to a specific category and the details of each category are the name of the category, description, and number of items are stored. Sometimes promotions are applied to the orders and the promotion details such as name of the promotion, discount applicable with the promotion, description, start date, and end date are stored along with the unique ID for each promotion

There are suppliers who supply the ingredients and each supplier is stored with the details such as name, address, contact number, and product category. All the ingredients are stored in the warehouse.

Chapter 2:

ER Diagram:



Data Dictionary:

The below table gives us the details of data that we used to construct our initial ER diagram.

Schema Construct	**Construct Meaning**	**Data Type**	**Structure and/or Constraints**
CUSTOMERS	Entity class to store information on Customers		
CustomerID	Unique identifier for each customer	NUMBER	Primary ID
Name	Composite Attribute. Contains FirstName and LastName	VARCHAR2(100 BYTE)	
Email	The email address of the customer	VARCHAR2(100 BYTE)	
Phone	The phone number of the customer for contact	VARCHAR2(20 BYTE)	
LOYALTY_PROGRAM	Entity class, to store information on Loyalty Program		
MembershipID	Unique identifier for each loyalty program	NUMBER	Primary ID
Description	A brief description of the loyalty program	VARCHAR2(255 BYTE)	
StartDate	The date when the loyalty program starts or was launched	DATE	
EndDate	The date when the loyalty program ends or expires	DATE	
RESERVATIONS	Entity class, to store information on Reservations		
ReservationID	Unique identifier for each Reservation	NUMBER	

ReservationTime	Time of the reservation	TIMESTAMP(6)	
PartySize	Number of people in reservation	NUMBER	
Preference	Preference of the table	VARCHAR2(255 BYTE)	
ContactPhone	Contact information	VARCHAR(15 BYTE)	
PROMOTIONS	Entity class, to store information on promotions		
PromotionsID	Unique identifier for each Promotion	NUMBER	Primary ID
PromotionName	Title of promotion	VARCHAR2(100 BYTE)	
Description	Brief description of promotion	VARCHAR2(255 BYTE)	
StartDate	Start date of promotion	DATE	
EndDate	End date of promotion	DATE	
Discount	Amount of Discount	NUMBER	
FEEDBACK	Entity class, to store information on customer feedback		
FeedbackID	Identifier	NUMBER	Primary ID
FeedbackDate	Date and time of feedback	DATE	
Rating	Scale of rating	NUMBER	
Comments	Multivalued composite attribute contains CommentID, Comment	VARCHAR2(255 BYTE)	
DELIVERY_PARTNER	Entity class, to store information on Delivery partner		
DeliveryPartnerID	Identifier	NUMBER	Primary ID
DeliveryPartnerName	Name of the Delivery Partner	VARCHAR2(100 BYTE)	
Email	Email address of the delivery	VARCHAR2(255 BYTE)	

	partner (Multivalued attribute)		
DeliveryMode	Type of Delivery (Express/ Normal)	VARCHAR2(50 BYTE)	
Address	Address of the delivery partner	VARCHAR2(255 BYTE)	
PrimaryPhone	Credit Limit of the delivery Partner	VARCHAR2(15 BYTE)	
PaymentMethod	The preferred payment method for the delivery parter	VARCHAR2(50 BYTE)	
DELIVERY	Entity class to store information on Delivery		
DeliveryID	Unique identifier for each delivery	NUMBER	Primary ID
DeliveryDate	The date and time when the delivery was scheduled or made	DATE	
DeliveryAddress	The address where the delivery is intended to be sent	VARCHAR2(255 BYTE)	
DeliveryStatus	The status of the delivery	VARCHAR2(50 BYTE)	
DeliveryDriver	The name or ID of the delivery driver responsible for the delivery	VARCHAR2(100 BYTE)	
DeliveryTime	Time at which order will be delivered	TIMESTAMP(6)	
ORDERS	Entity class to store information on Order Details		
OrderID	Unique identifier for each order	NUMBER	Primary ID
OrderDate	The date and time when the order was placed	DATE	
TotalAmount	The total cost of the order,	NUMBER	

	including all menu items		
PAYMENTS	Entity class to store information on Payment		
Payment ID	Unique identifier for each payment	NUMBER	Primary ID
PaymentMethod	The payment method used	VARCHAR2(50 BYTE)	
PaymentDate	The date and time when the payment was made	DATE	
PaymentAmount	The amount of the payment made for the order	NUMBER	
PaymentStatus	Status of the payment	VARCHAR2(50 BYTE)	
ORDER_DETAILS	Weak Entity class to store information on Order Details		
OrderItemReference	Partial identifier	NUMBER	
Item	Item Name that is ordered	VARCHAR2(100 BYTE)	
Quantity	Number of quantity ordered	NUMBER	
LOCATIONS	Entity class to store information on Location		
LocationID	Unique identifier for each location	NUMBER	Primary ID
Location Name	The name of the location	VARCHAR2(100 BYTE)	
Address	The address of the location, including street address, city, state, and postal code	VARCHAR2(255 BYTE)	
Phone	Multivalued attribute	VARCHAR2(20)	
EVENTS	Entity class, to store information on Events		
EventID	Unique identifier for each Event	NUMBER	Primary ID
EventName	Name of the Event	VARCHAR2(100 BYTE)	

Date	Date and Time of the Event	DATE	
Description	Details of the Event	VARCHAR2(255 BYTE)	
WAREHOUSE	Entity class, to store information on Warehouse		
WarehouseID	Unique identifier for each Warehouse	NUMBER	Primary ID
Address	Address of the warehouse	VARCHAR2(255 BYTE)	
Capacity	Capacity of the warehouse	NUMBER	
PhoneNum	Contact phone number of the warehouse	VARCHAR2(15 BYTE)	
WAREHOUSE_DETAILS	Weak Entity class to store information on Warehouse Details		
UnitsOfStocking	Number of units stocked	NUMBER	
DateOfStocking	When the item is stocked in warehouse	DATE	
MENU_ITEMS	Entity class to store information on Menu Item		
ItemId	Unique identifier for each menu item	NUMBER	Primary ID
Name	The name or title of the menu item	VARCHAR2(100 BYTE)	
Price	The price of the menu item	NUMBER	
Description	Decription of the item	VARCHAR2(255 BYTE)	
CATEGORIES	Entity class, to store information on Category		
CategoryID	Unique identifier for each category	NUMBER	Primary ID
CategoryName	Name of the Category	VARCHAR2(50 BYTE)	
NoOfItems	Number of items in the category	NUMBER	

Description	Description of the category	VARCHAR2(255 BYTE)	
INGREDIENTS	Entity class to store information on Ingredients		
IngredientID	Unique identifier for each ingredient	NUMBER	Primary ID
Ingredient Name	The name of the ingredient	VARCHAR2(100 BYTE)	
PricePerUnit	The price of the ingredient per unit of measurement	NUMBER	
UnitOfMeasurement	The unit of measurement for the ingredient	VARCHAR2(50 BYTE)	
SUPPLIERS	Entity class to store information on Supplier		
SupplierID	Unique identifier for each supplier	NUMBER	Primary ID
Name	The name or business name of the supplier	VARCHAR2(100 BYTE)	
ContactInfo	Contact information for the supplier, which may include an email address or phone number	VARCHAR2(100 BYTE)	
Address	The address where the supplier is located or can be reached	VARCHAR2(255 BYTE)	
ProductCategory	The category of products or ingredients supplied by the vendor	VARCHAR2(50 BYTE)	
EMPLOYEES	Entity class to store information on Employee		
EmployeeID	Unique identifier for each employee	NUMBER	Primary ID
FirstName	The first name of the employee	VARCHAR2(50 BYTE)	
LastName	The last name of the employee	VARCHAR2(50 BYTE)	

Position	The job position or role of the employee within the organization	VARCHAR2(50 BYTE)	
Salary	The salary or hourly wage of the employee	NUMBER	
MANAGER	Sub class to Employees. Stores information on Managers		
ManagerLocation	Store Manager Location	VARCHAR2(50 BYTE)	
DirectReportees	Number of employees reporting to manager	NUMBER	
STAFF	Sub class to Employees. Stores information on Staff		
ShiftSchedule	Work schedule of staff employees	VARCHAR2(50 BYTE)	

Chapter-3

Relational Design:

Normalization

BELONG_TO(ItemID, CategoryID)

Foreign Key(ItemID) references MENUITEMS

Foreign Key(CategoryID) references CATEGORIES

CANSUPPLY(ingredientID, SupplierID)

Foreign Key(ingredientID) references INGREDIENTS

Foreign Key(SupplierID) references SUPPLIERS

CATEGORIES(CategoryID, description, noOfItems, CategoryName)

CUSTOMERS(CustomerID, FirstName, LastName, Phone, Email)

CUSTOMER_MEMBERSHIP(CustomerID, membershipID)

Foreign Key(membershipID) references LOYALTY_PROGRAM

Foreign Key(CustomerID) references CUSTOMERS

DELIVERY (DeliveryID, DeliveryDate, DeliveryDriver, DeliveryAddress)

DELIVERY_ORDERDETAILS(DeliveryID, OrderID)

Foreign Key(OrderID) references ORDERS

Foreign Key (DeliveryID) references DELIVERY

DELIVERY_PARTNER(deliveryPartnerID, deliveryPartnername, deliverymode, payment_method, address, primaryPhone)

DELIVERY_PART_EMAIL(deliveryPartnerID, email)

Foreign Key(deliveryPartnerID) references DELIVERY_PARTNER

DELIVERY_DETAILS(DeliveryID, deliveryPartnerID, deliveryStatus, deliveryTime)

Foreign Key(deliveryPartnerID) references DELIVERY_PARTNER

Foreign Key(deliveryID) references DELIVERY

EMPLOYEES(employeeID, position, salary, firstName, lastName)

MANAGER(employeeID, branch, directReports)

Foreign Key(employeeID) references EMPLOYEES

MANAGER_REPORTEES(employeeID, reporteeID)

Foreign Key(employeeID) references EMPLOYEES

Foreign Key(reporteeID) references EMPLOYEES

STAFF(employeeID, shiftSchedule, jobTitle)

Foreign Key(employeeID) references EMPLOYEES

EVENTS(eventID, eventName, date, description)

FEEDBACK(feedbackId, rating, date)

FEEDBACK_ORDERDETAILS(feedbackId, OrderID)

Foreign Key(OrderID) references ORDERS

Foreign Key(feedbackId) references FEEDBACK

FEEDBACK_COMMENTS(feedbackId, CommentId, Comments)

Foreign Key(feedbackId) references FEEDBACK

HAS_INGREDIENTS(ingredientID, ItemID)

Foreign Key(ingredientID) references INGREDIENTS

Foreign Key(ItemID) references MENUITEMS

HOST_EVENTS(locationId, eventID)

Foreign Key(locationId) references LOCATIONS

Foreign Key(eventID) references EVENTS

INGREDIENTS(ingredientID, ingredientName, priceperunit, unitofmeasurement)

LOCATIONS(locationId, locationName, address)

LOCATION_MANAGER(locationId, managerID)

Foreign Key(locationId) references LOCATIONS

Foreign Key(managerID) references EMPLOYEES

LOC_PHONE(locationId, phone)

Foreign Key(locationId) references LOCATIONS

LOYALTY_PROGRAM(membershipID, description, startDate, endDate) – 5 entries

MENU_ITEMS(ItemID, Name, Price, Description)

ORDERS(OrderID, OrderDate, TotalAmount)

ORDER_PAYMENT(OrderID, paymentID)

ForeignKey(OrderID) references ORDERS

ForeignKey(paymentID) references PAYMENTS

ORDERS_CUSTOMERSDETAILS(OrderID, CustomerID)

ForeignKey(OrderID) references ORDERS

Foreign Key(CustomerID) references CUSTOMERS

ORDER_LOCATION(OrderID, LocationID)

ForeignKey(OrderID) references ORDERS

ForeignKey(locationID) references LOCATIONS

ORDER_DETAILS(ItemID, OrderID, OrderItemReference, Item, Quantity)

Foreign Key(ItemID) references MENUITEMS

Foreign Key(OrderID) references ORDERS

PAYMENTS(paymentId, paymentMethod, paymentStatus, paymentDate, paymentAmount)

PROMOTIONS (promotionID, promotionName, description, discount, startDate, endDate)

PROMOTION_ORDERDETAILS(promotionID, OrderID)

Foreign Key(promotionID) references PROMOTIONS

Foreign Key(OrderID) references ORDERS

RESERVATIONS(reservationID, reservationDate, reservationTime, partySize, preference, contactPhone)

RESERVATION_CUSTOMERDETAILS(reservationID, CustomerID)

Foreign Key(reservationID) references RESERVATIONS

Foreign Key(CustomerID) references CUSTOMERS

STORED_IN(ingredientId, warehouseId)

Foreign Key(ingredientId) references INGREDIENTS

Foreign Key(warehouseID) references warehouseID

SUPPLIERS(SupplierID, Name, Address, ContactInfo, ProductCategory)

WAREHOUSE(warehouseID, address, capacity, phoneNum)

WAREHOUSE_DETAILS(warehouseID, locationId, unitsOfStocking, dateOfStocking)

Foreign Key(locationId) references LOCATIONS

Foreign Key(warehouseID) references warehouseID

WORKS_AT(locationId, employeeID)

Foreign Key(locationId) references LOCATIONS

Foreign Key(employeeID) references EMPLOYEES

Data Dictionary:

Schema Construct	Datatype	Structure and/or Constraints
BELONG_TO		
ItemID	NUMBER	(Primary Key, Foreign Key referencing MENUITEMS)
CategoryID	NUMBER	Primary ID,
CANSUPPLY		
ingredientID	NUMBER	Primary Key, Foreign Key referencing INGREDIENTS
SupplierID	NUMBER	Primary Key, Foreign Key referencing SUPPLIERS
CATEGORIES		
CategoryID	NUMBER	Primary Key
description	VARCHAR2(50 BYTE)	
noOfItems	NUMBER	NOT NULL
CategoryName	VARCHAR2(50 BYTE)	UNIQUE
CUSTOMERS		
CustomerID	NUMBER	Primary Key
FirstName	VARCHAR2(100 BYTE)	NOT NULL
LastName	VARCHAR2(100 BYTE)	NOT NULL
Phone	VARCHAR2(20 BYTE)	Unique, NOT NULL
Email	VARCHAR2(100 BYTE)	Unique, NOT NULL

CUSTOMER_MEMBERSHIP		
CustomerID	NUMBER	Primary Key, Foreign Key referencing CUSTOMERS
membershipID	NUMBER	Primary Key, Foreign Key referencing LOYALTY_PROGRAM
DELIVERY		
DeliveryID	NUMBER	PRIMARY KEY
DeliveryDate	DATE	NOT NULL
DeliveryDriver	VARCHAR2(100 BYTE)	NOT NULL
DeliveryAddress	VARCHAR2(255 BYTE)	NOT NULL
DELIVERY_ORDERDETAILS		
DeliveryID	NUMBER	(Primary Key, Foreign Key referencing DELIVERY
OrderID	NUMBER	Primary Key, Foreign Key referencing ORDERS
DELIVERY_PARTNER		
deliveryPartnerID	NUMBER	Primary Key
deliveryPartnername	VARCHAR2(100 BYTE)	UNIQUE
deliverymode	VARCHAR2(50 BYTE)	NOT NULL
payment_method	VARCHAR2(20 BYTE)	NOT NULL
address	VARCHAR2(255 BYTE)	NOT NULL
primaryPhone	VARCHAR2(15 BYTE)	CHECK (LEN(primaryPhone) = 10
DELIVERY_PART_EMAIL		

deliveryPartnerID	NUMBER	Primary Key, Foreign Key referencing DELIVERY_PARTNER
email	VARCHAR2(20 BYTE)	CHECK (CHARINDEX('@', Email) > 1)
DELIVERY_DETAILS		
DeliveryID	NUMBER	Primary Key, Foreign Key referencing DELIVERY
deliveryPartnerID	NUMBER	Foreign Key referencing DELIVERY_PARTNER
deliveryStatus	VARCHAR2(20 BYTE)	CHECK (DeliveryStatus IN ('In Transit', 'Delivered', 'Delayed'))
deliveryTime	VARCHAR2(20 BYTE)	NOT NULL
EMPLOYEES		
employeeID	NUMBER	Primary Key
position	VARCHAR2(50 BYTE)	NOT NULL
salary	NOT NULL	CHECK (salary >= 0)
firstName	VARCHAR2(20 BYTE)	NOT NULL
lastName	VARCHAR2(20 BYTE)	NOT NULL
MANAGER		
employeeID	NUMBER	Primary Key, Foreign Key referencing EMPLOYEES
branch	VARCHAR2(30 BYTE)	NOT NULL
directReports	INT	CHECK (directReports >= 0)
MANAGER_REPORTEES		
employeeID	NUMBER	(Primary Key, Foreign Key referencing EMPLOYEES)
reporteeID	VARCHAR2(10 BYTE)	Primary Key, Foreign Key referencing EMPLOYEES)
STAFF		
employeeID	NUMBER	Primary Key, Foreign Key referencing

		EMPLOYEES
shiftSchedule	VARCHAR2(50 BYTE)	NOT NULL
jobTitle	VARCHAR2(50 BYTE)	NOT NULL
EVENTS		
eventID	NUMBER	Primary Key,
eventName	VARCHAR2(100 BYTE)	NOT NULL
date	DATE	NOT NULL
description	VARCHAR2(255 BYTE)	NOT NULL
FEEDBACK		
feedbackId	NUMBER	Primary Key
rating	NUMBER	CHECK (rating >= 1 AND rating <= 5)
date	DATE	Date, Format (MM-DD-YYYY)
FEEDBACK_ORDERDETAILS		
feedbackId	NUMBER	Primary Key, Foreign Key referencing FEEDBACK
OrderID	NUMBER	Primary Key, Foreign Key referencing ORDERS
FEEDBACK_COMMENTS		
feedbackId	NUMBER	Primary Key, Foreign Key referencing FEEDBACK
CommentId	NUMBER	Primary Key
Comments	VARCHAR2(255 BYTE)	
HAS_INGREDIENTS		
ingredientID	NUMBER	Primary Key
ItemID	NUMBER	Primary Key, Foreign Key referencing

		MENUITEMS
HOST_EVENTS		
locationId	NUMBER	Primary Key(Foreign Key referencing LOCATIONS)
eventID	NUMBER	Primary Key, Foreign Key referencing EVENTS
INGREDIENTS		
ingredientID	NUMBER	PRIMARY KEY
ingredientName	VARCHAR2(100 BYTE)	NOT NULL
priceperunit	NUMBER	CHECK (priceperunit >= 0)
unitofmeasurement	VARCHAR2(50 BYTE)	NOT NULL
LOCATIONS		
locationId	NUMBER	PRIMARY KEY
locationName	VARCHAR2(100 BYTE)	UNIQUE
address	VARCHAR2(255 BYTE)	NOT NULL
LOCATION_MANAGER		
locationId	NUMBER	Primary Key, Foreign Key referencing LOCATIONS
managerID	NUMBER	Primary Key, Foreign Key referencing EMPLOYEES
LOC_PHONE		
locationId	NUMBER	Primary Key, Foreign Key referencing LOCATIONS
phone	VARCHAR2(20 BYTE)	CHECK (LEN(primaryPhone) = 10
LOYALTY_PROGRAM		
membershipID	NUMBER	PRIMARY KEY

description	VARCHAR2(255 BYTE)	NOT NULL
startDate	DATE	NOT NULL
endDate	DATE	NOT NULL
MENU_ITEMS		
ItemID	NUMBER	PRIMARY KEY
Name	VARCHAR2(100 BYTE)	NOT NULL
Price	NUMBER	CHECK (Price >= 0) NOT NULL
Description	VARCHAR2(255 BYTE)	NOT NULL
ORDERS		
OrderID	NUMBER	PRIMARY KEY
OrderDate	DATE	NOT NULL
TotalAmount	NUMBER	CHECK (TotalAmount >= 0) NOT NULL
ORDER_PAYMENT		
OrderID	NUMBER	Primary Key, Foreign Key referencing ORDERS
paymentID	NUMBER	Primary Key, Foreign Key referencing PAYMENTS
ORDERS_CUSTOMERSDETAILS		
OrderID	NUMBER	Primary Key, Foreign Key referencing ORDERS
CustomerID	NUMBER	Primary Key, Foreign Key referencing CUSTOMERS
ORDER_LOCATION		
OrderID	NUMBER	Primary Key, Foreign Key referencing ORDERS
LocationID	NUMBER	(Primary Key, Foreign Key referencing

		LOCATIONS
ORDER_DETAILS		
ItemID	NUMBER	Primary Key, Foreign Key referencing MENUITEMS
OrderID	NUMBER	Primary Key, Foreign Key referencing ORDERS
OrderItemReference	NUMBER	NOT NULL
Item	VARCHAR2(100 BYTE)	NOT NULL
Quantity	NUMBER	CHECK (Quantity > 0) NOT NULL
PAYMENTS		
paymentId	NUMBER	PRIMARY KEY
paymentMethod	VARCHAR2(50 BYTE)	NOT NULL
paymentStatus	VARCHAR2(50 BYTE)	NOT NULL
paymentDate	DATE	NOT NULL
paymentAmount	NUMBER	CHECK (paymentAmount >= 0) NOT NULL
PROMOTIONS		
promotionID	NUMBER	PRIMARY KEY
promotionName	VARCHAR2(100 BYTE)	NOT NULL
description	VARCHAR2(255 BYTE)	NOR NULL
discount	NUMBER	CHECK (discount >= 0 AND discount <= 100) NOT NULL
startDate	DATE	NOT NULL
endDate	DATE	NOT NULL
PROMOTION_ORDERDETAILS		
promotionID	NUMBER	Primary Key, Foreign Key referencing

		PROMOTIONS
OrderID	NUMBER	Primary Key, Foreign Key referencing ORDERS
RESERVATIONS		
reservationID	NUMBER	PRIMARY KEY
reservationDate	DATE	NOT NULL
reservationTime	TIMESTAMP(6)	NOT NULL
partySize	NUMBER	CHECK (partySize > 0) NOT NULL
preference	VARCHAR2(255 BYTE)	
contactPhone	VARCHAR2(15 BYTE)	NOT NULL
RESERVATION_CUSTOMERDETAILS		
reservationID	NUMBER	(Primary Key, Foreign Key referencing RESERVATIONS)
CustomerID	NUMBER	Primary Key, Foreign Key referencing CUSTOMERS
STORED_IN		
ingredientId	NUMBER	Primary Key, Foreign Key referencing INGREDIENTS
warehouseId	NUMBER	Primary Key, Foreign Key referencing WAREHOUSE
SUPPLIERS		
SupplierID	NUMBER	PRIMARY KEY
Name	VARCHAR2(100 BYTE)	NOT NULL
Address	VARCHAR2(255 BYTE)	NOT NULL
ContactInfo	VARCHAR2(100 BYTE)	

ProductCategory	VARCHAR2(50 BYTE)	
WAREHOUSE		
warehouseID	NUMBER	PRIMARY KEY
address	VARCHAR2(255 BYTE)	
capacity	NUMBER	
phoneNum	VARCHAR2(15 BYTE)	
WAREHOUSE_DETAILS		
warehouseID	NUMBER	PRIMARY KEY
locationId	NUMBER	
unitsOfStocking	NUMBER	
dateOfStocking	DATE	
WORKS_AT		
locationId	NUMBER	Primary Key, Foreign Key referencing LOCATIONS
employeeID	NUMBER	Primary Key, Foreign Key referencing EMPLOYEES

Chapter 4:

Creating Tables:

- CREATE TABLE "MIS531GROUPS1C"."AUTH_GROUP"

```
(  "ID" NUMBER(11,0) GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1
MAXVALUE 99999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER
NOCYCLE  NOKEEP  NOSCALE  NOT NULL ENABLE,

      "NAME" NVARCHAR2(150),

       PRIMARY KEY ("ID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  TABLESPACE "USERS"  ENABLE,

       UNIQUE ("NAME")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  TABLESPACE "USERS"  ENABLE

   );
```

- CREATE TABLE "MIS531GROUPS1C"."AUTH_GROUP_PERMISSIONS"

```
(  "ID" NUMBER(19,0) GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1
MAXVALUE 9999999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER
NOCYCLE  NOKEEP  NOSCALE  NOT NULL ENABLE,

      "GROUP_ID" NUMBER(11,0) NOT NULL ENABLE,

      "PERMISSION_ID" NUMBER(11,0) NOT NULL ENABLE,

       PRIMARY KEY ("ID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  TABLESPACE "USERS"  ENABLE,

       CONSTRAINT "AUTH_GROU_GROUP_ID__0CD325B0_U" UNIQUE ("GROUP_ID",
"PERMISSION_ID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  TABLESPACE "USERS"  ENABLE,
```

```
       CONSTRAINT "AUTH_GROU_GROUP_ID_B120CBF9_F" FOREIGN KEY ("GROUP_ID")

        REFERENCES "MIS531GROUPS1C"."AUTH_GROUP" ("ID") DEFERRABLE INITIALLY
DEFERRED ENABLE,

       CONSTRAINT "AUTH_GROU_PERMISSIO_84C5C92E_F" FOREIGN KEY ("PERMISSION_ID")

        REFERENCES "MIS531GROUPS1C"."AUTH_PERMISSION" ("ID") DEFERRABLE INITIALLY
DEFERRED ENABLE

   );
```

- ```
  CREATE TABLE "MIS531GROUPS1C"."AUTH_PERMISSION"

   (  "ID" NUMBER(11,0) GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1
MAXVALUE 99999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER
NOCYCLE NOKEEP NOSCALE NOT NULL ENABLE,

      "NAME" NVARCHAR2(255),

      "CONTENT_TYPE_ID" NUMBER(11,0) NOT NULL ENABLE,

      "CODENAME" NVARCHAR2(100),

       PRIMARY KEY ("ID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)

  TABLESPACE "USERS" ENABLE,

       CONSTRAINT "AUTH_PERM_CONTENT_T_01AB375A_U" UNIQUE ("CONTENT_TYPE_ID",
"CODENAME")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)

  TABLESPACE "USERS" ENABLE,

       CONSTRAINT "AUTH_PERM_CONTENT_T_2F476E4B_F" FOREIGN KEY ("CONTENT_TYPE_ID")

        REFERENCES "MIS531GROUPS1C"."DJANGO_CONTENT_TYPE" ("ID") DEFERRABLE
INITIALLY DEFERRED ENABLE

   );
  ```
  ```

- CREATE TABLE "MIS531GROUPS1C"."AUTH_USER"

```
   (  "ID" NUMBER(11,0) GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1
MAXVALUE 99999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER
NOCYCLE  NOKEEP  NOSCALE  NOT NULL ENABLE,

      "PASSWORD" NVARCHAR2(128),

      "LAST_LOGIN" TIMESTAMP (6),

      "IS_SUPERUSER" NUMBER(1,0) NOT NULL ENABLE,

      "USERNAME" NVARCHAR2(150),

      "FIRST_NAME" NVARCHAR2(150),

      "LAST_NAME" NVARCHAR2(150),

      "EMAIL" NVARCHAR2(254),

      "IS_STAFF" NUMBER(1,0) NOT NULL ENABLE,

      "IS_ACTIVE" NUMBER(1,0) NOT NULL ENABLE,

      "DATE_JOINED" TIMESTAMP (6) NOT NULL ENABLE,

       CHECK ("IS_SUPERUSER" IN (0,1)) ENABLE,

       CHECK ("IS_STAFF" IN (0,1)) ENABLE,

       CHECK ("IS_ACTIVE" IN (0,1)) ENABLE,

       PRIMARY KEY ("ID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  TABLESPACE "USERS"  ENABLE,

       UNIQUE ("USERNAME")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  TABLESPACE "USERS"  ENABLE

   );
```

- CREATE TABLE "MIS531GROUPS1C"."AUTH_USER_GROUPS"

```
   (  "ID" NUMBER(19,0) GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1
MAXVALUE 99999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER
NOCYCLE  NOKEEP  NOSCALE  NOT NULL ENABLE,

      "USER_ID" NUMBER(11,0) NOT NULL ENABLE,
```

```
      "GROUP_ID" NUMBER(11,0) NOT NULL ENABLE,

       PRIMARY KEY ("ID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  TABLESPACE "USERS"  ENABLE,

       CONSTRAINT "AUTH_USER_USER_ID_G_94350C0C_U" UNIQUE ("USER_ID", "GROUP_ID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  TABLESPACE "USERS"  ENABLE,

       CONSTRAINT "AUTH_USER_USER_ID_6A12ED8B_F" FOREIGN KEY ("USER_ID")

        REFERENCES "MIS531GROUPS1C"."AUTH_USER" ("ID") DEFERRABLE INITIALLY
DEFERRED ENABLE,

       CONSTRAINT "AUTH_USER_GROUP_ID_97559544_F" FOREIGN KEY ("GROUP_ID")

        REFERENCES "MIS531GROUPS1C"."AUTH_GROUP" ("ID") DEFERRABLE INITIALLY
DEFERRED ENABLE

   );
```

- CREATE TABLE "MIS531GROUPS1C"."AUTH_USER_USER_PERMISSIONS"

```
   (  "ID" NUMBER(19,0) GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1
MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER
NOCYCLE  NOKEEP  NOSCALE  NOT NULL ENABLE,

      "USER_ID" NUMBER(11,0) NOT NULL ENABLE,

      "PERMISSION_ID" NUMBER(11,0) NOT NULL ENABLE,

       PRIMARY KEY ("ID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  TABLESPACE "USERS"  ENABLE,

       CONSTRAINT "AUTH_USER_USER_ID_P_14A6B632_U" UNIQUE ("USER_ID",
"PERMISSION_ID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  TABLESPACE "USERS"  ENABLE,

       CONSTRAINT "AUTH_USER_USER_ID_A95EAD1B_F" FOREIGN KEY ("USER_ID")

        REFERENCES "MIS531GROUPS1C"."AUTH_USER" ("ID") DEFERRABLE INITIALLY
DEFERRED ENABLE,

       CONSTRAINT "AUTH_USER_PERMISSIO_1FBB5F2C_F" FOREIGN KEY ("PERMISSION_ID")
```

```
        REFERENCES "MIS531GROUPS1C"."AUTH_PERMISSION" ("ID") DEFERRABLE INITIALLY
DEFERRED ENABLE
   );
```

-
```
   CREATE TABLE "MIS531GROUPS1C"."BELONG_TO"
   (  "ITEMID" NUMBER,
      "CATEGORYID" NUMBER,
       PRIMARY KEY ("ITEMID", "CATEGORYID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE,
       FOREIGN KEY ("ITEMID")
        REFERENCES "MIS531GROUPS1C"."MENU_ITEMS" ("ITEMID") ENABLE,
       FOREIGN KEY ("CATEGORYID")
        REFERENCES "MIS531GROUPS1C"."CATEGORIES" ("CATEGORYID") ENABLE
   );
```
-
```
  CREATE TABLE "MIS531GROUPS1C"."CANSUPPLY"
   (  "INGREDIENTID" NUMBER,
      "SUPPLIERID" NUMBER,
       PRIMARY KEY ("INGREDIENTID", "SUPPLIERID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE,
       FOREIGN KEY ("INGREDIENTID")
```

```
      REFERENCES "MIS531GROUPS1C"."INGREDIENTS" ("INGREDIENTID") ENABLE,
     FOREIGN KEY ("SUPPLIERID")
      REFERENCES "MIS531GROUPS1C"."SUPPLIERS" ("SUPPLIERID") ENABLE
 );
```

-
```
    CREATE TABLE "MIS531GROUPS1C"."CATEGORIES"
 (  "CATEGORYID" NUMBER,
    "DESCRIPTION" VARCHAR2(255 BYTE),
    "NOOFITEMS" NUMBER,
    "CATEGORYNAME" VARCHAR2(50 BYTE),
     PRIMARY KEY ("CATEGORYID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
 );
```

```
      CREATE TABLE "MIS531GROUPS1C"."CUSTOMER_MEMBERSHIP"
       (  "CUSTOMERID" NUMBER,
          "MEMBERSHIPID" NUMBER,
           PRIMARY KEY ("CUSTOMERID", "MEMBERSHIPID")
      USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
      STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
      PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
      BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
      TABLESPACE "USERS"  ENABLE,
           FOREIGN KEY ("MEMBERSHIPID")
```

```
        REFERENCES "MIS531GROUPS1C"."LOYALTY_PROGRAM" ("MEMBERSHIPID")
ENABLE,
       FOREIGN KEY ("CUSTOMERID")
        REFERENCES "MIS531GROUPS1C"."CUSTOMERS" ("CUSTOMERID") ENABLE
   );
```

- CREATE TABLE "MIS531GROUPS1C"."CUSTOMERS"

```
   (  "CUSTOMERID" NUMBER,
      "FIRSTNAME" VARCHAR2(50 BYTE),
      "LASTNAME" VARCHAR2(50 BYTE),
      "PHONE" VARCHAR2(15 BYTE),
      "EMAIL" VARCHAR2(255 BYTE),
       PRIMARY KEY ("CUSTOMERID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE);
```

- CREATE TABLE "MIS531GROUPS1C"."DELIVERY"

```
   (  "DELIVERYID" NUMBER,
      "DELIVERYDATE" DATE,
      "DELIVERYDRIVER" VARCHAR2(100 BYTE),
      "DELIVERYADDRESS" VARCHAR2(255 BYTE),
      "DELIVERYSTATUS" VARCHAR2(50 BYTE),
      "DELIVERYTIME" TIMESTAMP (6),
       PRIMARY KEY ("DELIVERYID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
```

```
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
 );
```

- ```
  CREATE TABLE "MIS531GROUPS1C"."DELIVERY_ORDERDETAILS"
  ```

```
 (  "DELIVERYID" NUMBER,
    "ORDERID" NUMBER,
     PRIMARY KEY ("DELIVERYID", "ORDERID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
     FOREIGN KEY ("ORDERID")
      REFERENCES "MIS531GROUPS1C"."ORDERS" ("ORDERID") ENABLE,
     FOREIGN KEY ("DELIVERYID")
      REFERENCES "MIS531GROUPS1C"."DELIVERY" ("DELIVERYID") ENABLE
 );
```

- ```
  CREATE TABLE "MIS531GROUPS1C"."DELIVERY_PART_EMAIL"
  ```

```
 (  "DELIVERYPARTNERID" NUMBER,
    "EMAIL" VARCHAR2(255 BYTE),
     PRIMARY KEY ("DELIVERYPARTNERID", "EMAIL")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
     FOREIGN KEY ("DELIVERYPARTNERID")
```

```
        REFERENCES "MIS531GROUPS1C"."DELIVERY_PARTNER" ("DELIVERYPARTNERID")
ENABLE

   );
```

- CREATE TABLE "MIS531GROUPS1C"."DELIVERY_PARTNER"

```
   (  "DELIVERYPARTNERID" NUMBER,

      "DELIVERYPARTNERNAME" VARCHAR2(100 BYTE),

      "DELIVERYMODE" VARCHAR2(50 BYTE),

      "PAYMENTMETHOD" VARCHAR2(50 BYTE),

      "ADDRESS" VARCHAR2(255 BYTE),

      "PRIMARYPHONE" VARCHAR2(15 BYTE),

       PRIMARY KEY ("DELIVERYPARTNERID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)

  TABLESPACE "USERS"  ENABLE

   );
```

- CREATE TABLE "MIS531GROUPS1C"."DJANGO_ADMIN_LOG"

```
   (  "ID" NUMBER(11,0) GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1
MAXVALUE 99999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20
NOORDER  NOCYCLE  NOKEEP  NOSCALE  NOT NULL ENABLE,

      "ACTION_TIME" TIMESTAMP (6) NOT NULL ENABLE,

      "OBJECT_ID" NCLOB,

      "OBJECT_REPR" NVARCHAR2(200),

      "ACTION_FLAG" NUMBER(11,0) NOT NULL ENABLE,

      "CHANGE_MESSAGE" NCLOB,

      "CONTENT_TYPE_ID" NUMBER(11,0),

      "USER_ID" NUMBER(11,0) NOT NULL ENABLE,

       CHECK ("ACTION_FLAG" >= 0) ENABLE,
```

```
       PRIMARY KEY ("ID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  TABLESPACE "USERS"  ENABLE,

       CONSTRAINT "DJANGO_AD_CONTENT_T_C4BCE8EB_F" FOREIGN KEY
("CONTENT_TYPE_ID")

        REFERENCES "MIS531GROUPS1C"."DJANGO_CONTENT_TYPE" ("ID") DEFERRABLE
INITIALLY DEFERRED ENABLE,

       CONSTRAINT "DJANGO_AD_USER_ID_C564EBA6_F" FOREIGN KEY ("USER_ID")

        REFERENCES "MIS531GROUPS1C"."AUTH_USER" ("ID") DEFERRABLE INITIALLY
DEFERRED ENABLE

   );
```

- CREATE TABLE "MIS531GROUPS1C"."DJANGO_CONTENT_TYPE"

```
   (  "ID" NUMBER(11,0) GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1
MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20
NOORDER  NOCYCLE  NOKEEP  NOSCALE  NOT NULL ENABLE,

      "APP_LABEL" NVARCHAR2(100),

      "MODEL" NVARCHAR2(100),

       PRIMARY KEY ("ID")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)

  TABLESPACE "USERS"  ENABLE,

       CONSTRAINT "DJANGO_CO_APP_LABEL_76BD3D3B_U" UNIQUE ("APP_LABEL",
"MODEL")

  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)

  TABLESPACE "USERS"  ENABLE

   );
```

- CREATE TABLE "MIS531GROUPS1C"."DJANGO_MIGRATIONS"

("ID" NUMBER(19,0) GENERATED BY DEFAULT ON NULL AS IDENTITY MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL ENABLE,

"APP" NVARCHAR2(255),

"NAME" NVARCHAR2(255),

"APPLIED" TIMESTAMP (6) NOT NULL ENABLE,

PRIMARY KEY ("ID")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)

TABLESPACE "USERS" ENABLE

);

- CREATE TABLE "MIS531GROUPS1C"."DJANGO_SESSION"

("SESSION_KEY" NVARCHAR2(40) NOT NULL ENABLE,

"SESSION_DATA" NCLOB,

"EXPIRE_DATE" TIMESTAMP (6) NOT NULL ENABLE,

PRIMARY KEY ("SESSION_KEY")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS

TABLESPACE "USERS" ENABLE

);

- CREATE TABLE "MIS531GROUPS1C"."EMPLOYEES"

("EMPLOYEEID" NUMBER,

"POSITION" VARCHAR2(50 BYTE),

"SALARY" NUMBER,

"FIRSTNAME" VARCHAR2(50 BYTE),

"LASTNAME" VARCHAR2(50 BYTE),

PRIMARY KEY ("EMPLOYEEID")

```
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
 );
```

- ```
  CREATE TABLE "MIS531GROUPS1C"."EVENTS"
  ```

```
 (  "EVENTID" NUMBER,
    "EVENTNAME" VARCHAR2(100 BYTE),
    "EVENTDATE" DATE,
    "EVENTDESCRIPTION" VARCHAR2(255 BYTE),
     PRIMARY KEY ("EVENTID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
 );
```

- ```
  CREATE TABLE "MIS531GROUPS1C"."FEEDBACK"
  ```

```
 (  "FEEDBACKID" NUMBER,
    "RATING" NUMBER,
    "FEEDBACKDATE" DATE,
     PRIMARY KEY ("FEEDBACKID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
```

```
);
```

- ```
  CREATE TABLE "MIS531GROUPS1C"."FEEDBACK_COMMENTS"
  ```

```
(  "FEEDBACKID" NUMBER,
   "COMMENTID" NUMBER,
   "COMMENTS" VARCHAR2(255 BYTE),
    PRIMARY KEY ("FEEDBACKID", "COMMENTID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
    FOREIGN KEY ("FEEDBACKID")
     REFERENCES "MIS531GROUPS1C"."FEEDBACK" ("FEEDBACKID") ENABLE
);
```

- ```
  CREATE TABLE "MIS531GROUPS1C"."FEEDBACK_ORDERDETAILS"
  ```

```
(  "FEEDBACKID" NUMBER,
   "ORDERID" NUMBER,
    PRIMARY KEY ("FEEDBACKID", "ORDERID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
    FOREIGN KEY ("FEEDBACKID")
     REFERENCES "MIS531GROUPS1C"."FEEDBACK" ("FEEDBACKID") ENABLE,
    FOREIGN KEY ("ORDERID")
     REFERENCES "MIS531GROUPS1C"."ORDERS" ("ORDERID") ENABLE
);
```

```
● CREATE TABLE "MIS531GROUPS1C"."HAS_INGREDIENTS"
(  "INGREDIENTID" NUMBER,
   "ITEMID" NUMBER,
    PRIMARY KEY ("INGREDIENTID", "ITEMID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
    FOREIGN KEY ("INGREDIENTID")
     REFERENCES "MIS531GROUPS1C"."INGREDIENTS" ("INGREDIENTID") ENABLE,
    FOREIGN KEY ("ITEMID")
     REFERENCES "MIS531GROUPS1C"."MENU_ITEMS" ("ITEMID") ENABLE
);
● CREATE TABLE "MIS531GROUPS1C"."HOST_EVENTS"
(  "LOCATIONID" NUMBER,
   "EVENTID" NUMBER,
    PRIMARY KEY ("LOCATIONID", "EVENTID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
    FOREIGN KEY ("LOCATIONID")
     REFERENCES "MIS531GROUPS1C"."LOCATIONS" ("LOCATIONID") ENABLE,
    FOREIGN KEY ("EVENTID")
     REFERENCES "MIS531GROUPS1C"."EVENTS" ("EVENTID") ENABLE
);
```

- CREATE TABLE "MIS531GROUPS1C"."INGREDIENTS"

```
(  "INGREDIENTID" NUMBER,
   "INGREDIENTNAME" VARCHAR2(100 BYTE),
   "PRICEPERUNIT" NUMBER,
   "UNITOFMEASUREMENT" VARCHAR2(50 BYTE),
    PRIMARY KEY ("INGREDIENTID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
);
```

- CREATE TABLE "MIS531GROUPS1C"."LOC_PHONE"

```
(  "LOCATIONID" NUMBER,
   "PHONE" VARCHAR2(15 BYTE),
    PRIMARY KEY ("LOCATIONID", "PHONE")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
    FOREIGN KEY ("LOCATIONID")
     REFERENCES "MIS531GROUPS1C"."LOCATIONS" ("LOCATIONID") ENABLE
);
```

- CREATE TABLE "MIS531GROUPS1C"."LOCATION_MANAGER"

```
(  "LOCATIONID" NUMBER,
   "MANAGERID" NUMBER,
    PRIMARY KEY ("LOCATIONID", "MANAGERID")
```

```
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
     FOREIGN KEY ("LOCATIONID")
      REFERENCES "MIS531GROUPS1C"."LOCATIONS" ("LOCATIONID") ENABLE,
     FOREIGN KEY ("MANAGERID")
      REFERENCES "MIS531GROUPS1C"."EMPLOYEES" ("EMPLOYEEID") ENABLE
 );
```

- CREATE TABLE "MIS531GROUPS1C"."LOCATIONS"

```
 (  "LOCATIONID" NUMBER,
    "LOCATIONNAME" VARCHAR2(100 BYTE),
    "ADDRESS" VARCHAR2(255 BYTE),
     PRIMARY KEY ("LOCATIONID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
 );
```

- CREATE TABLE "MIS531GROUPS1C"."LOYALTY_PROGRAM"

```
 (  "MEMBERSHIPID" NUMBER,
    "DESCRIPTION" VARCHAR2(255 BYTE),
    "STARTDATE" DATE,
    "ENDDATE" DATE,
     PRIMARY KEY ("MEMBERSHIPID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
```

```
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
 );
```

- CREATE TABLE "MIS531GROUPS1C"."MANAGER"

```
 (  "EMPLOYEEID" NUMBER,
    "MANAGERLOCATION" VARCHAR2(50 BYTE),
    "DIRECTREPORTS" NUMBER,
     PRIMARY KEY ("EMPLOYEEID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
     FOREIGN KEY ("EMPLOYEEID")
      REFERENCES "MIS531GROUPS1C"."EMPLOYEES" ("EMPLOYEEID") ENABLE
 );
```

- CREATE TABLE "MIS531GROUPS1C"."MANAGER_REPORTEES"

```
 (  "EMPLOYEEID" NUMBER,
    "REPORTEEID" NUMBER,
     PRIMARY KEY ("EMPLOYEEID", "REPORTEEID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
     FOREIGN KEY ("EMPLOYEEID")
```

```
      REFERENCES "MIS531GROUPS1C"."EMPLOYEES" ("EMPLOYEEID") ENABLE,
     FOREIGN KEY ("REPORTEEID")
      REFERENCES "MIS531GROUPS1C"."EMPLOYEES" ("EMPLOYEEID") ENABLE
 );
```

- CREATE TABLE "MIS531GROUPS1C"."MENU_ITEMS"

```
 (  "ITEMID" NUMBER,
    "NAME" VARCHAR2(100 BYTE),
    "PRICE" NUMBER,
    "DESCRIPTION" VARCHAR2(255 BYTE),
     PRIMARY KEY ("ITEMID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
 );
```

- CREATE TABLE "MIS531GROUPS1C"."ORDER_DETAILS"

```
 (  "ITEMID" NUMBER,
    "ORDERID" NUMBER,
    "ORDERITEMREFERENCE" NUMBER,
    "ITEM" VARCHAR2(100 BYTE),
    "QUANTITY" NUMBER,
     PRIMARY KEY ("ITEMID", "ORDERID", "ORDERITEMREFERENCE")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
```

```
     FOREIGN KEY ("ITEMID")
      REFERENCES "MIS531GROUPS1C"."MENU_ITEMS" ("ITEMID") ENABLE,
     FOREIGN KEY ("ORDERID")
      REFERENCES "MIS531GROUPS1C"."ORDERS" ("ORDERID") ENABLE
 );
```

- CREATE TABLE "MIS531GROUPS1C"."ORDER_LOCATION"

```
 (  "ORDERID" NUMBER,
    "LOCATIONID" NUMBER,
     PRIMARY KEY ("ORDERID", "LOCATIONID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
     FOREIGN KEY ("ORDERID")
      REFERENCES "MIS531GROUPS1C"."ORDERS" ("ORDERID") ENABLE,
     FOREIGN KEY ("LOCATIONID")
      REFERENCES "MIS531GROUPS1C"."LOCATIONS" ("LOCATIONID") ENABLE
 );
```

- CREATE TABLE "MIS531GROUPS1C"."ORDER_PAYMENT"

```
 (  "ORDERID" NUMBER,
    "PAYMENTID" NUMBER,
     PRIMARY KEY ("ORDERID", "PAYMENTID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
```

```
     FOREIGN KEY ("ORDERID")
      REFERENCES "MIS531GROUPS1C"."ORDERS" ("ORDERID") ENABLE,
     FOREIGN KEY ("PAYMENTID")
      REFERENCES "MIS531GROUPS1C"."PAYMENTS" ("PAYMENTID") ENABLE
 );
```

-
```
 CREATE TABLE "MIS531GROUPS1C"."ORDERS"
 (  "ORDERID" NUMBER,
    "ORDERDATE" DATE,
    "TOTALAMOUNT" NUMBER,
     PRIMARY KEY ("ORDERID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
 );
```

-
```
 CREATE TABLE "MIS531GROUPS1C"."PAYMENTS"
 (  "PAYMENTID" NUMBER,
    "PAYMENTMETHOD" VARCHAR2(50 BYTE),
    "PAYMENTSTATUS" VARCHAR2(50 BYTE),
    "PAYMENTDATE" DATE,
    "PAYMENTAMOUNT" NUMBER,
     PRIMARY KEY ("PAYMENTID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
```

```
);
```

- CREATE TABLE "MIS531GROUPS1C"."PROMOTION_ORDERDETAILS"

```
(  "PROMOTIONID" NUMBER,
   "ORDERID" NUMBER,
    PRIMARY KEY ("PROMOTIONID", "ORDERID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
    FOREIGN KEY ("PROMOTIONID")
     REFERENCES "MIS531GROUPS1C"."PROMOTIONS" ("PROMOTIONID") ENABLE,
    FOREIGN KEY ("ORDERID")
     REFERENCES "MIS531GROUPS1C"."ORDERS" ("ORDERID") ENABLE
);
```

- CREATE TABLE "MIS531GROUPS1C"."PROMOTIONS"

```
(  "PROMOTIONID" NUMBER,
   "PROMOTIONNAME" VARCHAR2(100 BYTE),
   "DESCRIPTION" VARCHAR2(255 BYTE),
   "DISCOUNT" NUMBER,
   "STARTDATE" DATE,
   "ENDDATE" DATE,
    PRIMARY KEY ("PROMOTIONID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
```

```
);
```

- ```
  CREATE TABLE "MIS531GROUPS1C"."RESERVATION_CUSTOMERDETAILS"
  ```

```
(  "RESERVATIONID" NUMBER,
   "CUSTOMERID" NUMBER,
    PRIMARY KEY ("RESERVATIONID", "CUSTOMERID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
    FOREIGN KEY ("RESERVATIONID")
     REFERENCES "MIS531GROUPS1C"."RESERVATIONS" ("RESERVATIONID") ENABLE,
    FOREIGN KEY ("CUSTOMERID")
     REFERENCES "MIS531GROUPS1C"."CUSTOMERS" ("CUSTOMERID") ENABLE
);
```

- ```
  CREATE TABLE "MIS531GROUPS1C"."RESERVATIONS"
  ```

```
(  "RESERVATIONID" NUMBER,
   "RESERVATIONTIME" TIMESTAMP (6),
   "PARTYSIZE" NUMBER,
   "PREFERENCE" VARCHAR2(255 BYTE),
   "CONTACTPHONE" VARCHAR2(15 BYTE),
    PRIMARY KEY ("RESERVATIONID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
);
```

```
● CREATE TABLE "MIS531GROUPS1C"."STAFF"
 (  "EMPLOYEEID" NUMBER,
    "SHIFTSCHEDULE" VARCHAR2(50 BYTE),
     PRIMARY KEY ("EMPLOYEEID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
     FOREIGN KEY ("EMPLOYEEID")
      REFERENCES "MIS531GROUPS1C"."EMPLOYEES" ("EMPLOYEEID") ENABLE
 );
● CREATE TABLE "MIS531GROUPS1C"."STORED_IN"
 (  "INGREDIENTID" NUMBER,
    "WAREHOUSEID" NUMBER,
     PRIMARY KEY ("INGREDIENTID", "WAREHOUSEID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
     FOREIGN KEY ("INGREDIENTID")
      REFERENCES "MIS531GROUPS1C"."INGREDIENTS" ("INGREDIENTID") ENABLE,
     FOREIGN KEY ("WAREHOUSEID")
      REFERENCES "MIS531GROUPS1C"."WAREHOUSE" ("WAREHOUSEID") ENABLE
 );
● CREATE TABLE "MIS531GROUPS1C"."SUPPLIERS"
 (  "SUPPLIERID" NUMBER,
```

```
    "NAME" VARCHAR2(100 BYTE),
    "ADDRESS" VARCHAR2(255 BYTE),
    "CONTACTINFO" VARCHAR2(100 BYTE),
    "PRODUCTCATEGORY" VARCHAR2(50 BYTE),
     PRIMARY KEY ("SUPPLIERID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE
 );
```

- ```
  CREATE TABLE "MIS531GROUPS1C"."WAREHOUSE"
  (  "WAREHOUSEID" NUMBER,
     "ADDRESS" VARCHAR2(255 BYTE),
     "CAPACITY" NUMBER,
     "PHONENUM" VARCHAR2(15 BYTE),
      PRIMARY KEY ("WAREHOUSEID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"  ENABLE
  );
  ```
- ```
  CREATE TABLE "MIS531GROUPS1C"."WAREHOUSE_DETAILS"
  (  "WAREHOUSEID" NUMBER,
     "LOCATIONID" NUMBER,
     "UNITSOFSTOCKING" NUMBER,
     "DATEOFSTOCKING" DATE,
  ```

```
     PRIMARY KEY ("WAREHOUSEID", "LOCATIONID", "DATEOFSTOCKING")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
     FOREIGN KEY ("WAREHOUSEID")
      REFERENCES "MIS531GROUPS1C"."WAREHOUSE" ("WAREHOUSEID") ENABLE,
     FOREIGN KEY ("LOCATIONID")
      REFERENCES "MIS531GROUPS1C"."LOCATIONS" ("LOCATIONID") ENABLE
 );
```

-
```
 CREATE TABLE "MIS531GROUPS1C"."WORKS_AT"
 (  "LOCATIONID" NUMBER,
    "EMPLOYEEID" NUMBER,
     PRIMARY KEY ("LOCATIONID", "EMPLOYEEID")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS"  ENABLE,
     FOREIGN KEY ("LOCATIONID")
      REFERENCES "MIS531GROUPS1C"."LOCATIONS" ("LOCATIONID") ENABLE,
     FOREIGN KEY ("EMPLOYEEID")
      REFERENCES "MIS531GROUPS1C"."EMPLOYEES" ("EMPLOYEEID") ENABLE
 );
```

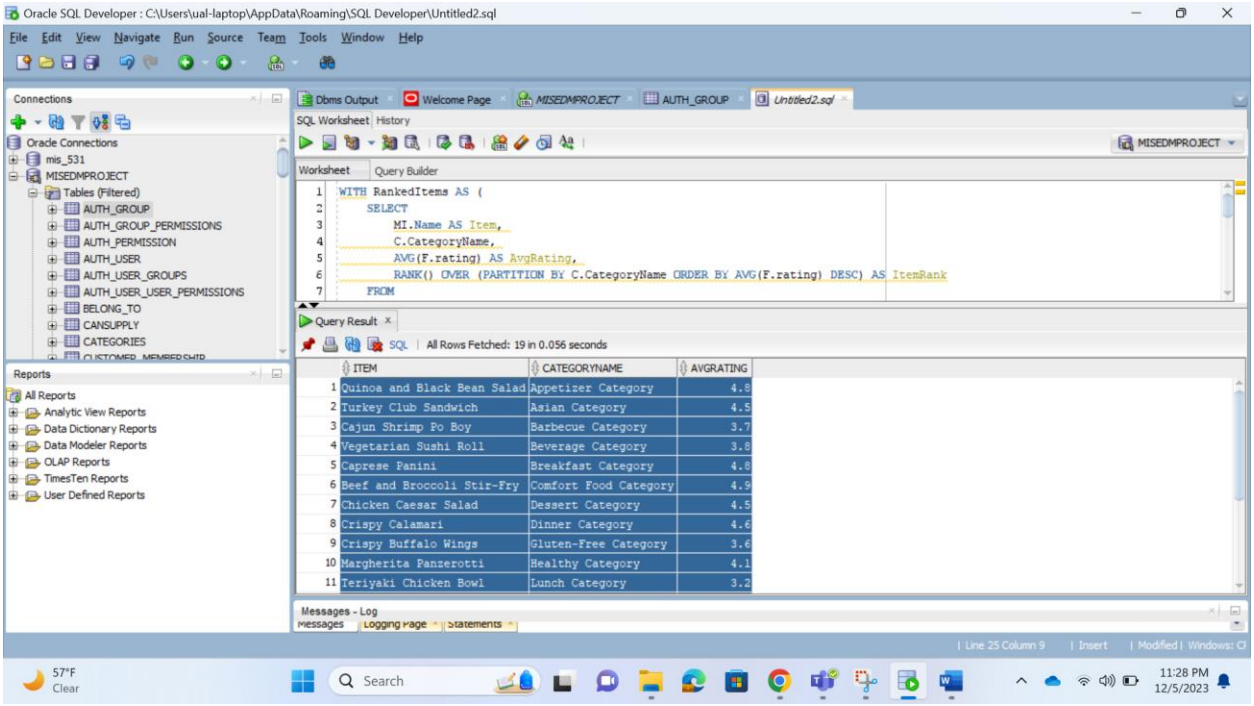
SQL Queries:

Query 1: Highest-rated item in every category

```
WITH RankedItems AS (
    SELECT
        MI.Name AS Item,
        C.CategoryName,
        AVG(F.rating) AS AvgRating,
        RANK() OVER (PARTITION BY C.CategoryName ORDER BY AVG(F.rating) DESC) AS
ItemRank
    FROM
        FEEDBACK F
        JOIN FEEDBACK_ORDERDETAILS FO ON F.feedbackid = FO.feedbackid
        JOIN ORDER_DETAILS OD ON FO.orderid = OD.orderid
        JOIN BELONG_TO BT ON OD.itemid = BT.itemid
        JOIN CATEGORIES C ON BT.categoryid = C.categoryid
        JOIN MENU_ITEMS MI ON BT.itemid = MI.itemid
    GROUP BY
        MI.Name, C.CategoryName
)
SELECT
    Item,
    CategoryName,
    AvgRating
FROM
    RankedItems
WHERE
    ItemRank = 1
```

```
ORDER BY
    CategoryName;
```

Output:



Explanation:

- The query aims to find the highest-rated item in each category based on customer feedback.
- It uses a CTE (RankedItems) to calculate the average rating and rank for each item within its category.
- The main query then filters the results to include only items with a rank of 1 (the highest rating) within each category.
- The final result includes the item name, category name, and average rating, ordered by category.

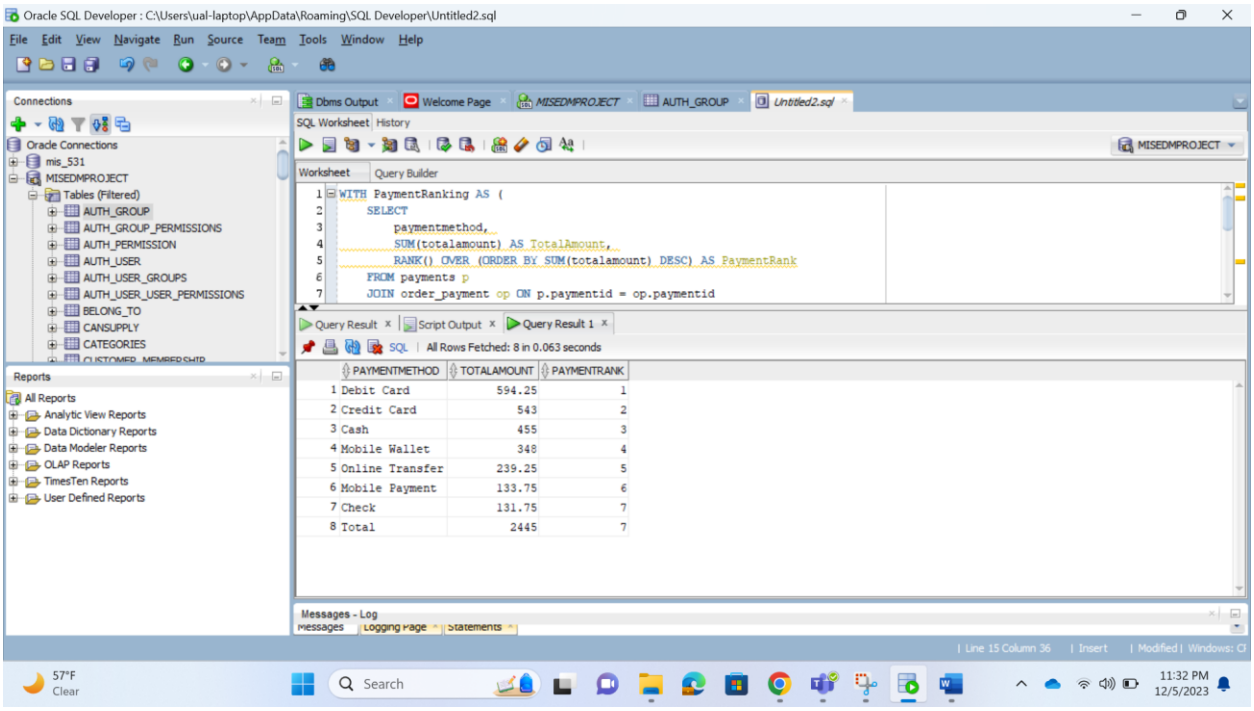
Query 2: Breakdown of the total amount for each payment method, along with subtotals for each distinct payment method and a grand total.

```
WITH PaymentRanking AS (
    SELECT
```

```
        paymentmethod,
        SUM(totalamount) AS TotalAmount,
        RANK() OVER (ORDER BY SUM(totalamount) DESC) AS PaymentRank
    FROM payments p
    JOIN order_payment op ON p.paymentid = op.paymentid
    JOIN orders o ON op.orderid = o.orderid
    GROUP BY paymentmethod
)

SELECT
    COALESCE(paymentmethod, 'Total') AS PaymentMethod,
    SUM(TotalAmount) AS TotalAmount,
    MAX(PaymentRank) AS PaymentRank
FROM PaymentRanking
GROUP BY ROLLUP(paymentmethod)
ORDER BY PaymentRank;
```

Output:



Explanation:

- The query aims to provide a breakdown of the total amount for each payment method, along with subtotals for each distinct payment method and a grand total.
- It uses a CTE (PaymentRanking) to calculate the total amount and rank for each distinct payment method based on orders and payments.
- The main query then uses the COALESCE function to replace NULL values in the paymentmethod column with 'Total'.
- The ROLLUP grouping set is used to calculate subtotals and the grand total.
- The final result includes the payment method, total amount, and payment rank for each grouping, ordered by payment rank.

Query 3: Top supplier for each ingredient

```
SELECT ingredientName, Name, SupplierRank

FROM (

    SELECT

        ingredientName,

        Name,

        RANK() OVER (PARTITION BY ingredientName ORDER BY Frequency DESC) AS
SupplierRank

    FROM (

        SELECT

            I.ingredientName,

            S.Name,

            COUNT(*) AS Frequency

        FROM CANSUPPLY CS

        JOIN INGREDIENTS I ON CS.ingredientID = I.ingredientID

        JOIN SUPPLIERS S ON CS.SupplierID = S.SupplierID

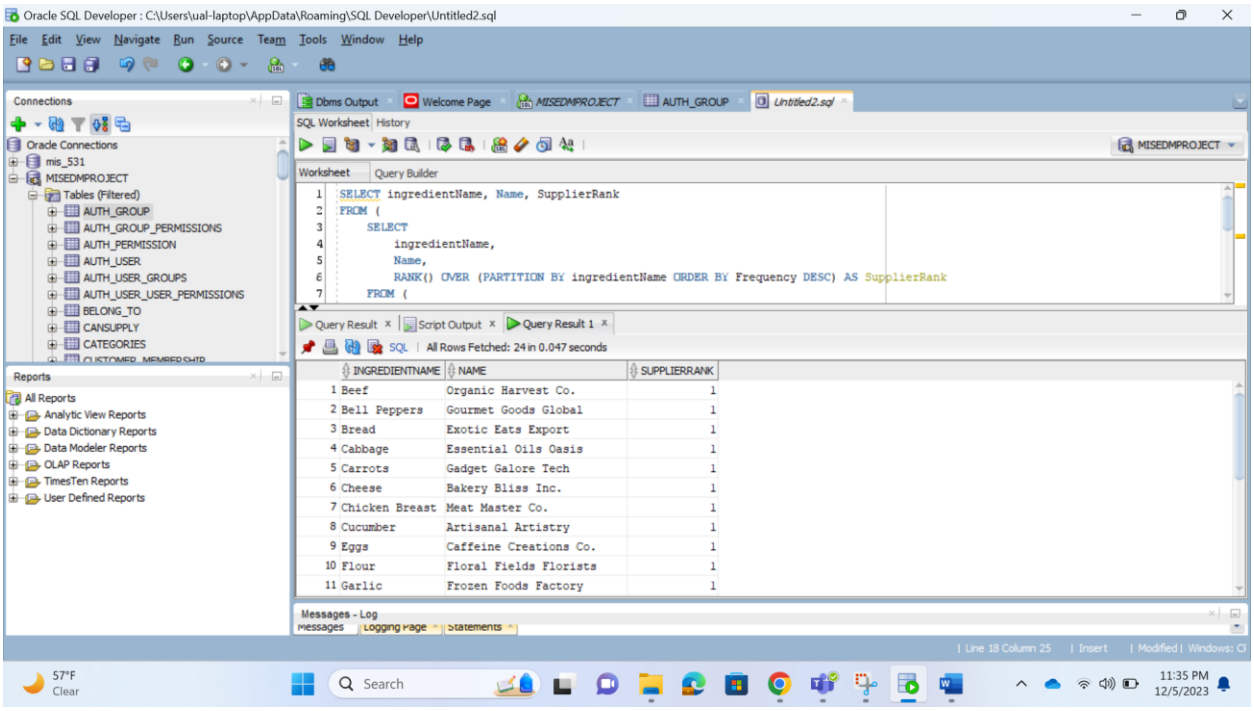
        GROUP BY I.ingredientName, S.Name

    )

) ranked_data

WHERE SupplierRank <= 1;
```

Output:



Explanation:

- The query aims to find the top supplier for each ingredient based on the frequency of occurrences in the CANSUPPLY table.
- The innermost query retrieves the count of occurrences for each combination of ingredient and supplier.
- The ranked data subquery calculates the rank of each supplier for each ingredient based on the frequency.
- The main query selects the ingredient name, supplier name, and supplier rank, filtering to include only the top supplier for each ingredient.

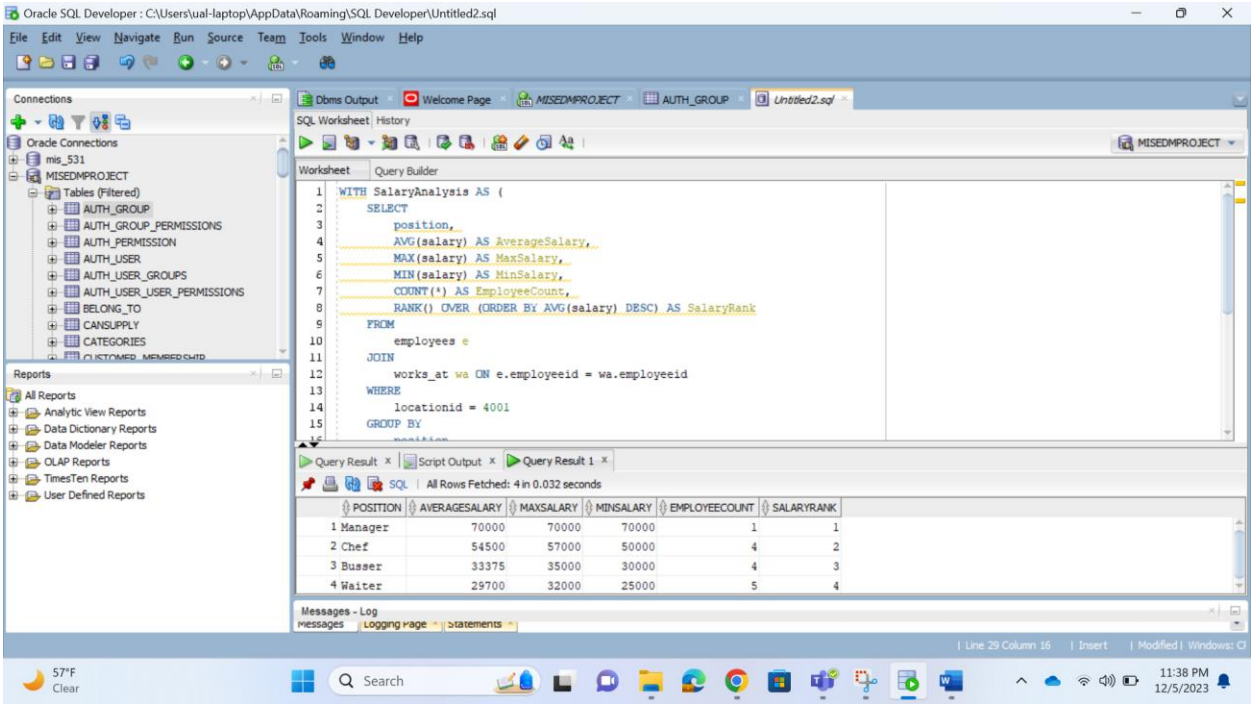
Query 4: For each category of jobTitle display salary statistics.

```
WITH SalaryAnalysis AS (
    SELECT
        position,
        AVG(salary) AS AverageSalary,
        MAX(salary) AS MaxSalary,
        MIN(salary) AS MinSalary,
        COUNT(*) AS EmployeeCount,
        RANK() OVER (ORDER BY AVG(salary) DESC) AS SalaryRank
    FROM
        employees e
    JOIN
        works_at wa ON e.employeeid = wa.employeeid
    WHERE
        locationid = 4001
    GROUP BY
        position
)

SELECT
    position,
    AverageSalary,
    MaxSalary,
    MinSalary,
    EmployeeCount,
    SalaryRank
FROM
    SalaryAnalysis
ORDER BY
```

```
SalaryRank;
```

Output:



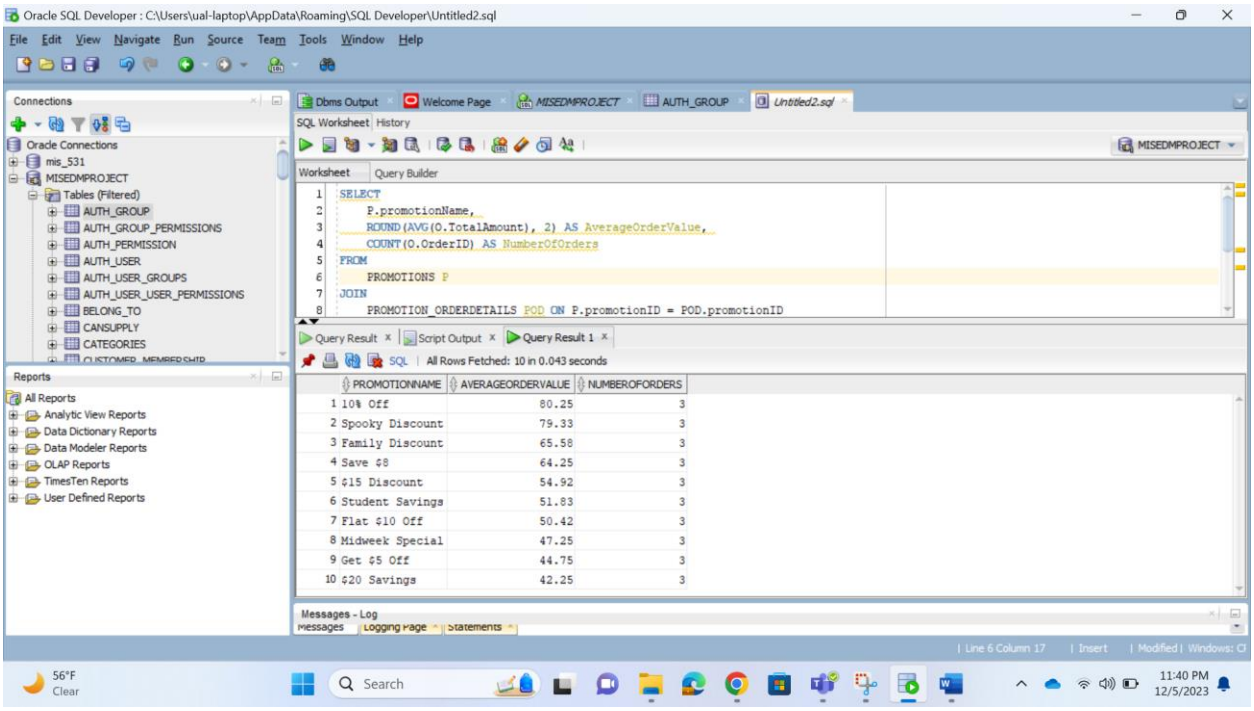
Explanation:

- The query aims to display salary statistics for each category of job title (position).
- The CTE (SalaryAnalysis) calculates average, maximum, and minimum salaries, along with the employee count for each job position at a specific location.
- The main query selects the relevant columns from the CTE and orders the result by the salary rank.

Query 5: Provide info on promotions by calculating the average order value (rounded to two decimal places) and the number of orders for each promotion.

```
SELECT
    P.promotionName,
    ROUND(AVG(O.TotalAmount), 2) AS AverageOrderValue,
    COUNT(O.OrderID) AS NumberOfOrders
FROM
    PROMOTIONS P
JOIN
    PROMOTION_ORDERDETAILS POD ON P.promotionID = POD.promotionID
JOIN
    ORDERS O ON POD.OrderID = O.OrderID
GROUP BY
    P.promotionName
ORDER BY
    AverageOrderValue DESC;
```

Output:



Explanation:

- The query calculates the average order value and counts the number of orders for each promotion.
- It retrieves the promotion name, the rounded average order value, and the number of orders for each promotion.
- The results are grouped by promotion name and ordered in descending order based on the average order value.

Query 6: Summary of deliveries based on the day of the week, including the number of deliveries and the total value of orders for each day.

```
WITH DeliverySummary AS (

    SELECT

        TRIM(TO_CHAR(D.DeliveryDate, 'Day')) AS Weekday,

        COUNT(D.DeliveryID) AS NumberOfDeliveries,

        SUM(O.TotalAmount) AS TotalValueOfOrders

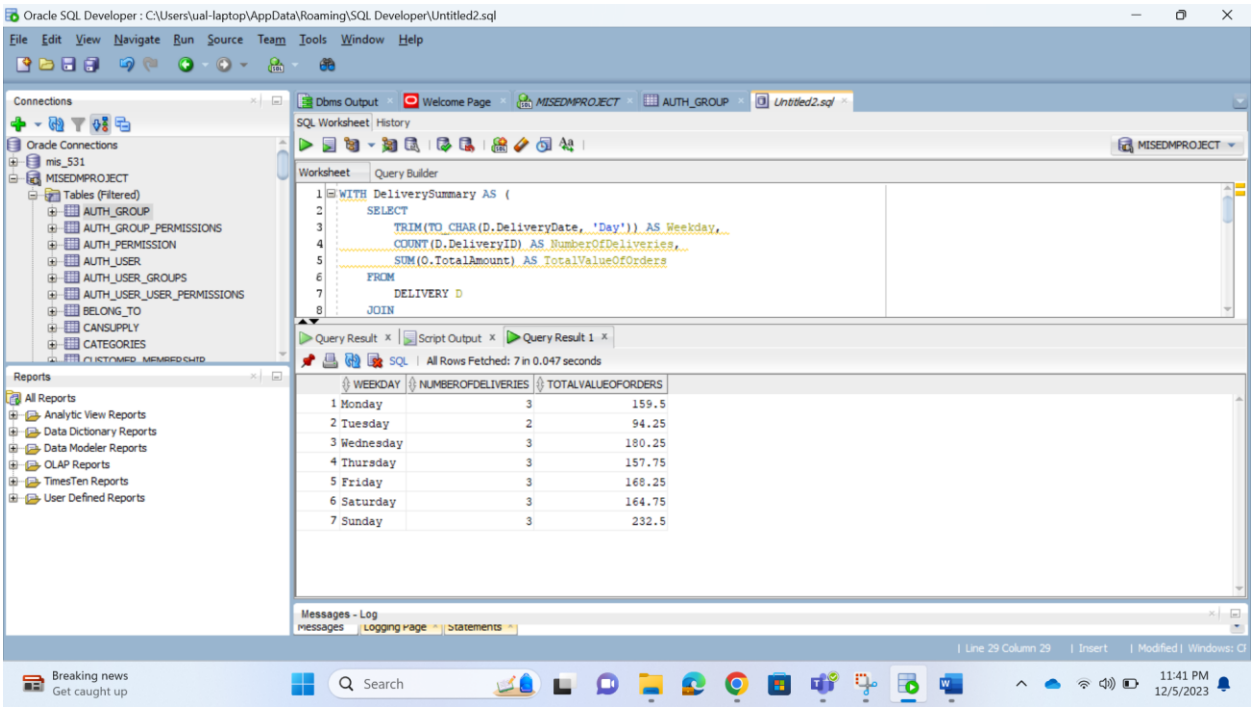
    FROM

        DELIVERY D

    JOIN
```

```
        DELIVERY_ORDERDETAILS DO ON D.DeliveryID = DO.DeliveryID
    JOIN
        ORDERS O ON DO.OrderID = O.OrderID
    GROUP BY
        TRIM(TO_CHAR(D.DeliveryDate, 'Day'))
)
SELECT
    Weekday,
    NumberOfDeliveries,
    TotalValueOfOrders
FROM
    DeliverySummary
ORDER BY
    CASE Weekday
        WHEN 'Monday' THEN 1
        WHEN 'Tuesday' THEN 2
        WHEN 'Wednesday' THEN 3
        WHEN 'Thursday' THEN 4
        WHEN 'Friday' THEN 5
        WHEN 'Saturday' THEN 6
        WHEN 'Sunday' THEN 7
    END;
```

Output:



Explanation:

- The query aims to provide a summary of deliveries based on the day of the week.
- The CTE (DeliverySummary) calculates the number of deliveries and the total value of orders for each day of the week.
- The main query selects the weekday, number of deliveries, and total value of orders from the CTE.
- The results are ordered based on the specified order of weekdays.

Query 7: Summary of deliveries based on the day of the week, including the number of deliveries and the total value of orders for each day.

```
SELECT

    CASE

        WHEN mi.Price < 10 THEN 'Affordable'

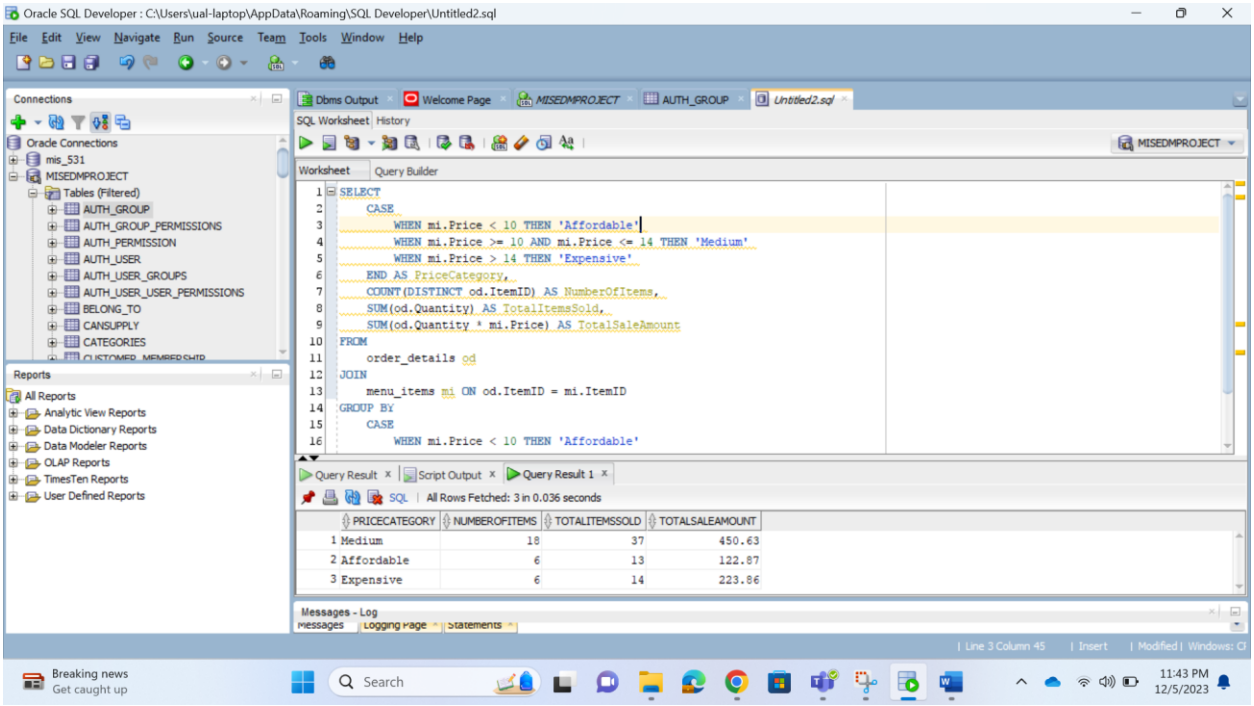
        WHEN mi.Price >= 10 AND mi.Price <= 14 THEN 'Medium'

        WHEN mi.Price > 14 THEN 'Expensive'

    END AS PriceCategory,
```

```
    COUNT(DISTINCT od.ItemID) AS NumberOfItems,
    SUM(od.Quantity) AS TotalItemsSold,
    SUM(od.Quantity * mi.Price) AS TotalSaleAmount
FROM
    order_details od
JOIN
    menu_items mi ON od.ItemID = mi.ItemID
GROUP BY
    CASE
        WHEN mi.Price < 10 THEN 'Affordable'
        WHEN mi.Price >= 10 AND mi.Price <= 14 THEN 'Medium'
        WHEN mi.Price > 14 THEN 'Expensive'
    END;
```

Output:



Explanation:

- The query provides a summary of sales based on different price categories for menu items.
- The CASE statement categorizes items into 'Affordable,' 'Medium,' and 'Expensive' based on their prices.
- It then calculates the number of distinct menu items sold, the total quantity of items sold, and the total sale amount for each price category.
- The results are grouped by the assigned price category, providing insights into sales performance across different price ranges.

Query 8: Most recent stocking information in each warehouse.

```
WITH LatestStock AS (

    SELECT

        w.warehouseID,

        w.address AS WarehouseAddress,

        hi.ingredientID,

        i.ingredientName,

        wd.unitsOfStocking,

        wd.dateOfStocking,

        ROW_NUMBER() OVER (PARTITION BY w.warehouseID, hi.ingredientID ORDER BY
wd.dateOfStocking DESC) AS StockingRank

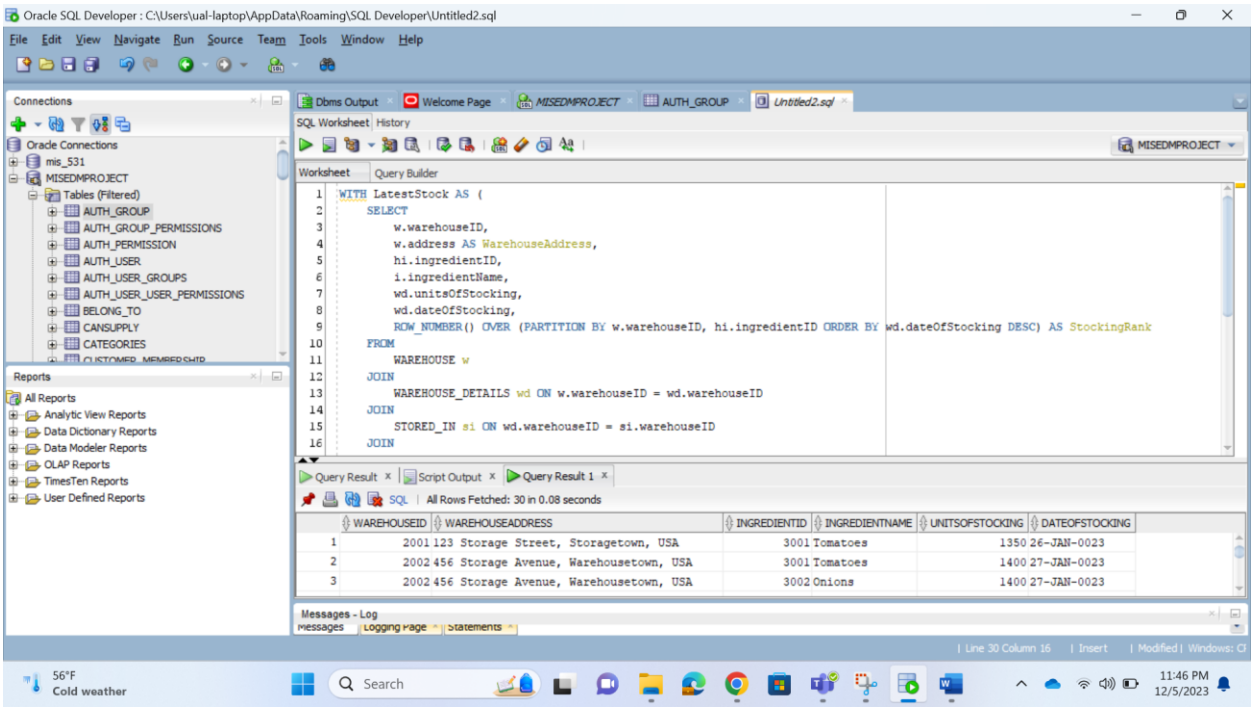
    FROM

        WAREHOUSE w
```

```
    JOIN
        WAREHOUSE_DETAILS wd ON w.warehouseID = wd.warehouseID
    JOIN
        STORED_IN si ON wd.warehouseID = si.warehouseID
    JOIN
        HAS_INGREDIENTS hi ON si.ingredientID = hi.ingredientID
    JOIN
        INGREDIENTS i ON hi.ingredientID = i.ingredientID
)

SELECT
    warehouseID,
    WarehouseAddress,
    ingredientID,
    ingredientName,
    unitsOfStocking,
    dateOfStocking
FROM
    LatestStock
WHERE
    StockingRank = 1;
```

Output:



Explanation:

- The query aims to retrieve the most recent stocking information for each warehouse and ingredient combination.
- The CTE (LatestStock) uses the ROW_NUMBER() window function to assign a rank to each stocking entry within the same warehouse and ingredient partition, ordered by the descending date of stocking.
- The main query selects columns from the CTE where the StockingRank is equal to 1, ensuring that only the most recent stocking entry for each warehouse and ingredient is included in the final result.

Query 9: Cumulative sales

```
SELECT

    o.OrderDate,

    SUM(o.TotalAmount) AS DailySales,

    SUM(SUM(o.TotalAmount)) OVER (ORDER BY o.OrderDate) AS CumulativeSales

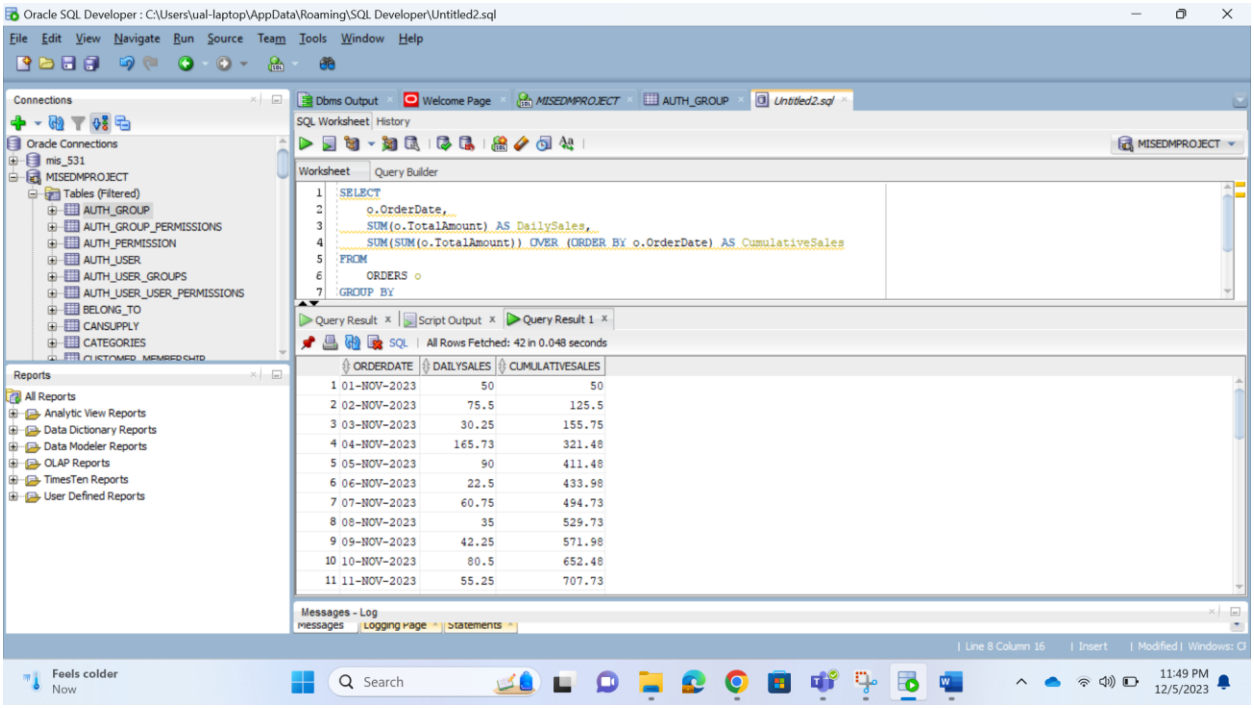
FROM

    ORDERS o

GROUP BY
```

```
    o.OrderDate
ORDER BY
    o.OrderDate;
```

Output:



Explanation:

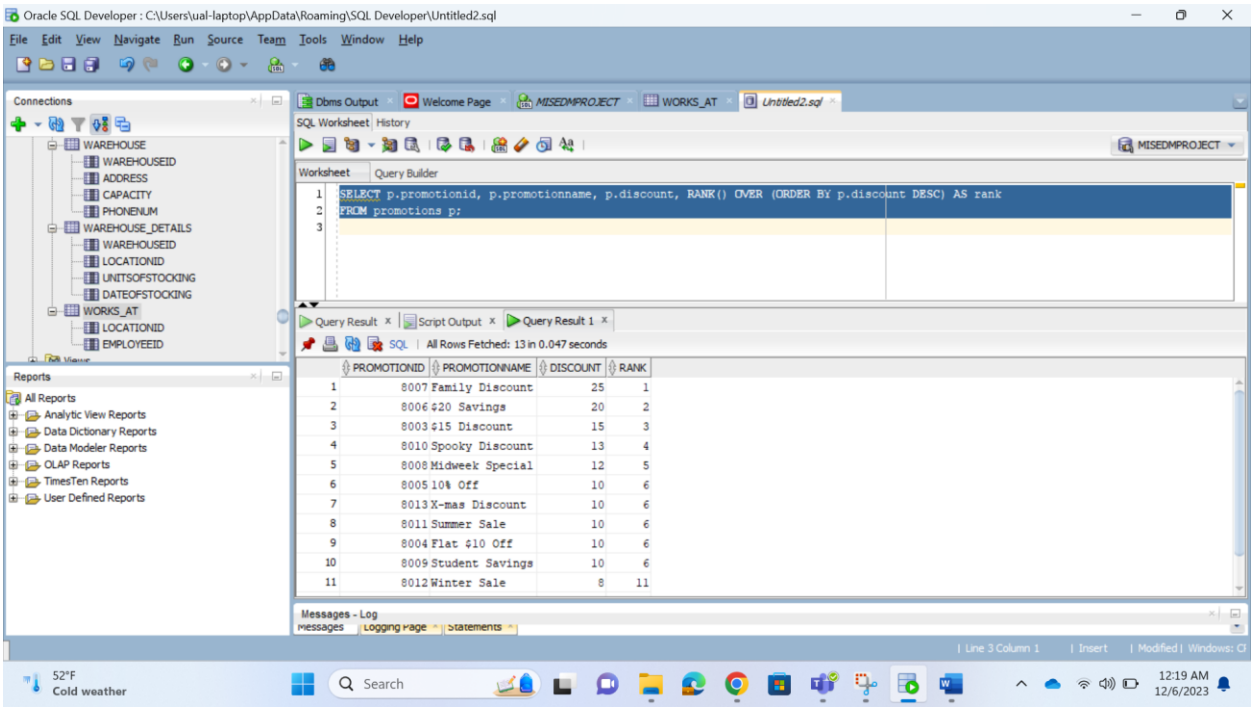
- The query calculates daily and cumulative sales for each order date.
- SUM(o.TotalAmount) AS DailySales calculates the total sales amount for each day.
- SUM(SUM(o.TotalAmount)) OVER (ORDER BY o.OrderDate) AS CumulativeSales uses the window function SUM() to calculate the cumulative sales. The ORDER BY clause in the window function specifies the order of summation based on the OrderDate.

Query 10: Rank the promotions based on their discount percentages in descending order.

```
SELECT p.promotionid, p.promotionname, p.discount, RANK() OVER (ORDER BY p.discount
DESC) AS rank

FROM promotions p;
```

Output:



Explanation:

- The query retrieves information about promotions from the promotions table.
- It includes the promotionid, promotionname, and discount columns.
- The RANK() OVER (ORDER BY p.discount DESC) AS rank calculates the rank of each promotion based on its discount percentage in descending order.

Query 11: Top 3 Categories by the Number of Items with Ranks

```
SELECT

    categoryname,

    total_items,

    RANK() OVER (ORDER BY total_items DESC) AS category_rank

FROM (

    SELECT c.categoryname, COUNT(b.itemid) AS total_items

    FROM categories c

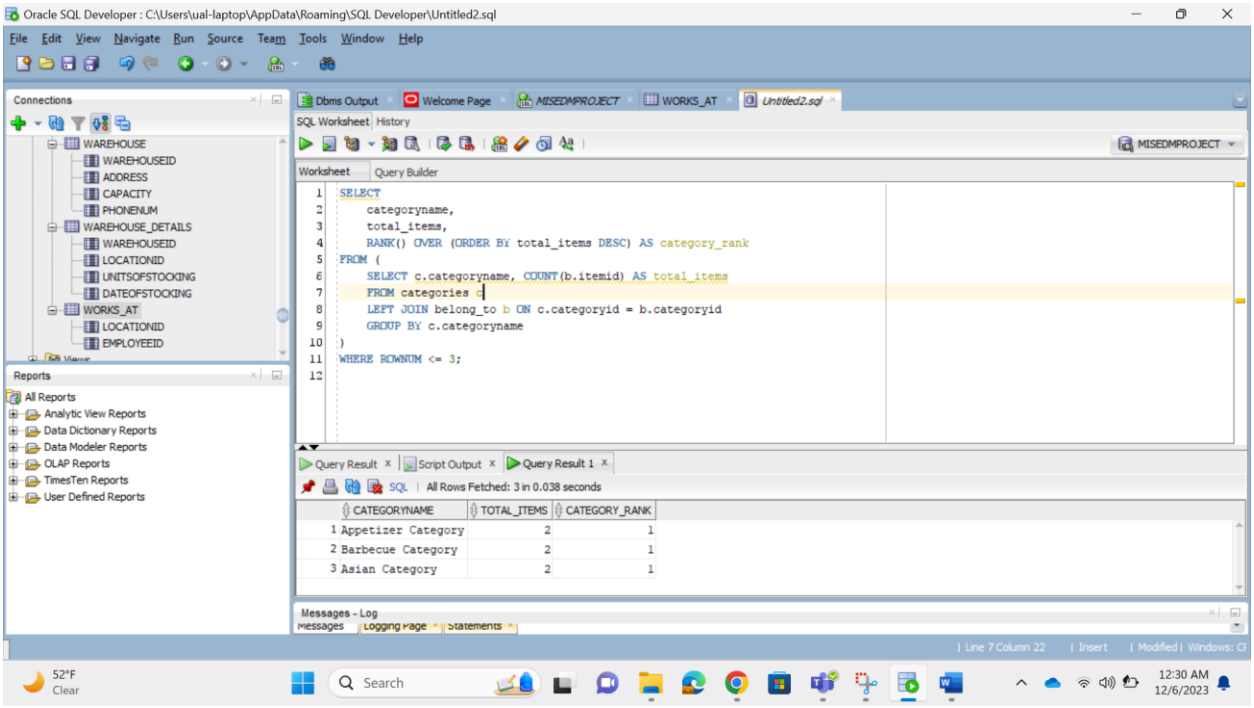
    LEFT JOIN belong_to b ON c.categoryid = b.categoryid

    GROUP BY c.categoryname
```

```
)

WHERE ROWNUM <= 3;
```

Output:



Explanation:

- The query aims to find the top 3 categories by the number of items and assign ranks to them.
- The inner query calculates the total number of items for each category using a LEFT JOIN between the categories and belong_to tables.
- The outer query then selects the category name, total items, and assigns ranks based on the total number of items in descending order.
- The WHERE ROWNUM <= 3 ensures that only the top 3 categories are included in the final result.

Query 12: Retrieve the total count of events hosted at each location, including the rank of locations by event count

```
SELECT l.locationid, l.locationname, COUNT(he.eventid) AS event_count,

       RANK() OVER (ORDER BY COUNT(he.eventid) DESC) AS rank_by_event_count

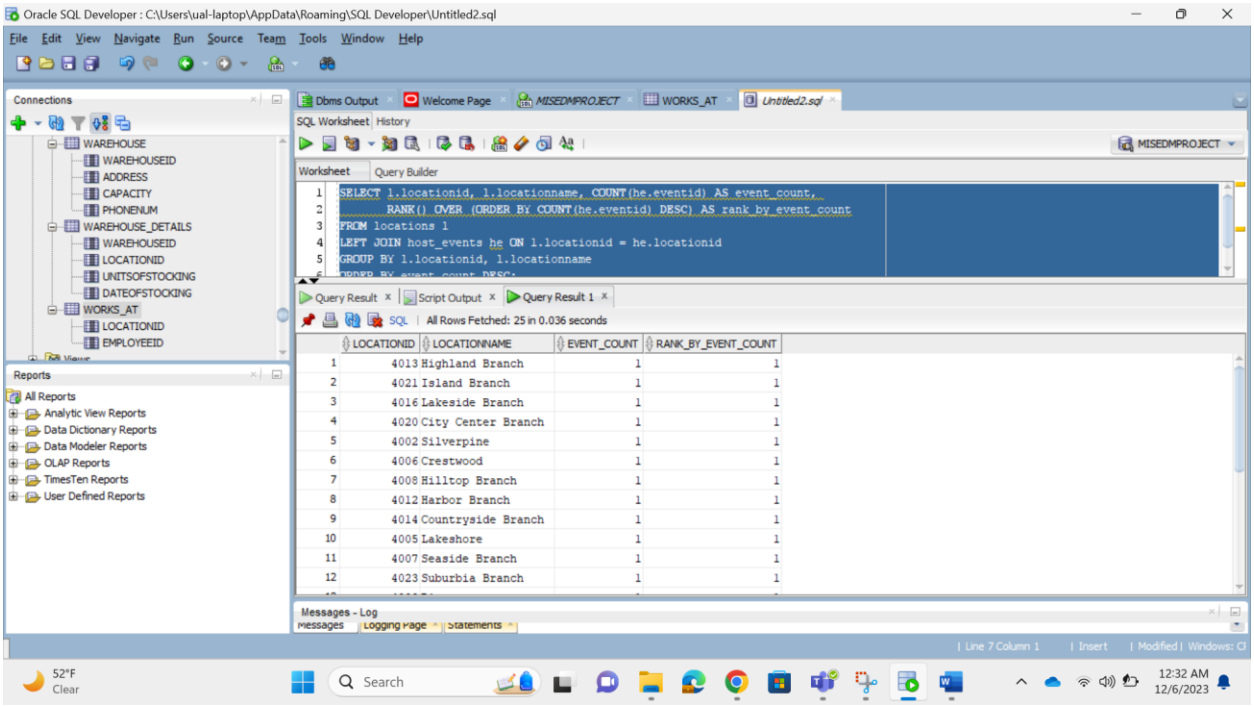
FROM locations l
```

```
LEFT JOIN host_events he ON l.locationid = he.locationid

GROUP BY l.locationid, l.locationname

ORDER BY event_count DESC;
```

Output:



Explanation:

- The query retrieves information about the total count of events hosted at each location and assigns ranks to locations based on the event count.
- The LEFT JOIN is used to include all locations, even those with zero events.
- The GROUP BY clause groups the results by locationid and locationname to calculate the count of events for each location.
- The RANK() OVER (ORDER BY COUNT(he.eventid) DESC) calculates the rank of each location based on the count of events in descending order.
- The final result set includes columns for locationid, locationname, event_count, and the assigned rank (rank_by_event_count), ordered by the event count in descending order.

Chapter 5:

Triggers and Procedures

Procedure: Procedure for Calculating Total Sales by MenuItem and Updating Description

```
CREATE OR REPLACE PROCEDURE CalcTotalSalesByMenuItem (startDate DATE, endDate DATE)
 IS
      CURSOR menu_sales_cursor IS
      SELECT mi.ItemID, mi.Name, SUM(od.Quantity) as TotalSales
      FROM MENU_ITEMS mi
      JOIN ORDER_DETAILS od ON mi.ItemID = od.ItemID
      JOIN ORDERS o ON od.OrderID = o.OrderID
      WHERE o.OrderDate BETWEEN startDate AND endDate
      GROUP BY mi.ItemID, mi.Name;

      rec menu_sales_cursor%ROWTYPE;
 BEGIN
      OPEN menu_sales_cursor;
      LOOP
      FETCH menu_sales_cursor INTO rec;
      EXIT WHEN menu_sales_cursor%NOTFOUND;

      -- Output total sales for each menu item
      DBMS_OUTPUT.PUT_LINE('Item: ' || rec.Name || ', Total Sales: ' ||
rec.TotalSales);



      -- Example: Update the DESCRIPTION column based on total sales
      IF rec.TotalSales > 1000 THEN
            UPDATE MENU_ITEMS
            SET DESCRIPTION = 'High Sales Item'
            WHERE ItemID = rec.ItemID;
      ELSE
            UPDATE MENU_ITEMS
            SET DESCRIPTION = 'Regular Sales Item'
            WHERE ItemID = rec.ItemID;
      END IF;
      END LOOP;
      CLOSE menu_sales_cursor;
 END CalcTotalSalesByMenuItem;



DECLARE
      startDate DATE := TO_DATE('2023-01-01', 'YYYY-MM-DD'); -- Replace with your
```

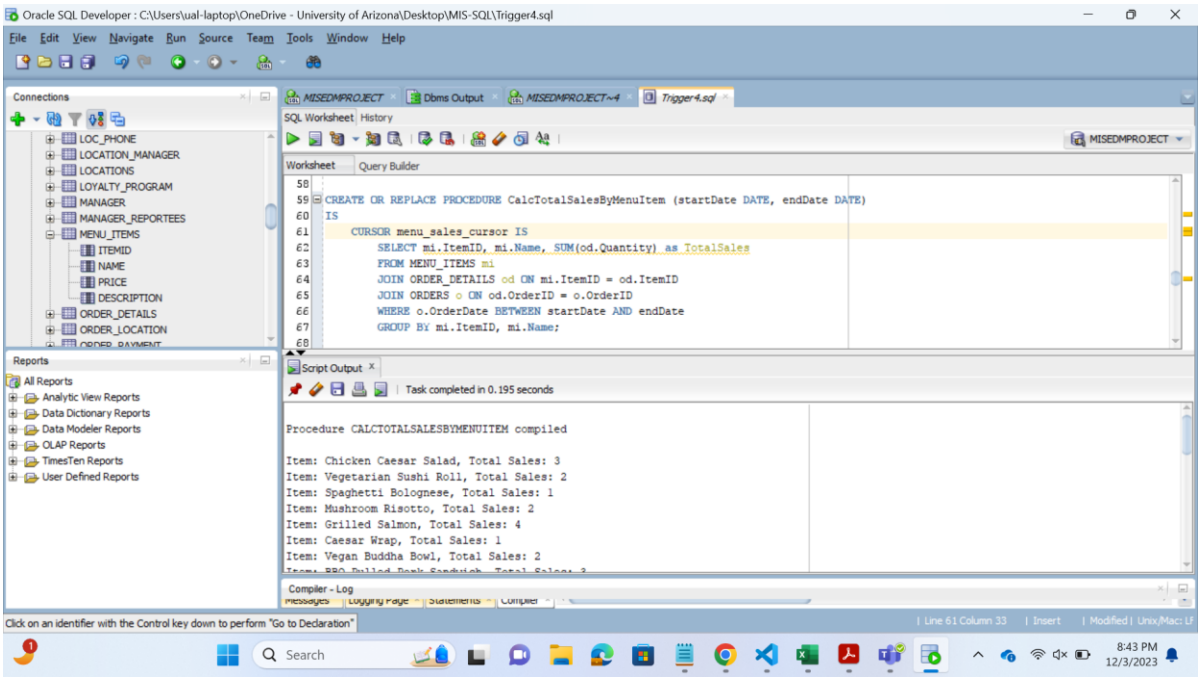
```
desired start date
      endDate DATE := TO_DATE('2023-12-31', 'YYYY-MM-DD');    -- Replace with your
desired end date
 BEGIN
      CalcTotalSalesByMenuItem(startDate, endDate);
 END;
 /
```

Explanation:

- The procedure calculates the total sales for each menu item within a specified date range and outputs the information.
- It also provides an example of updating the DESCRIPTION column of the MENU_ITEMS table based on the total sales.
- Adjust the startDate and endDate values in the DECLARE block based on your desired date range.

In natural language, the functionality of this procedure is to analyze the total sales of each menu item within a specified date range, print the results to the console, and update the description of each menu item based on its total sales. The trigger is helpful because it automates the process of analyzing and categorizing menu items based on their sales performance, providing insights into the popularity and revenue contribution of each item. This can be valuable for business decision-making, marketing strategies, and menu optimization in a restaurant or similar business context.

Output:



Procedure: Procedure for Retrieving Main Ingredients and Managing Ingredients Table

```
CREATE SEQUENCE ingredient_id_sequence START WITH 1 INCREMENT BY 1;



CREATE OR REPLACE PROCEDURE main_ingredients_of_order (order_id
ORDERS.OrderID%TYPE)
 IS
      CURSOR ingredient_cursor IS
      SELECT ingredientName
      FROM INGREDIENTS
      JOIN HAS_INGREDIENTS ON INGREDIENTS.ingredientID =
HAS_INGREDIENTS.ingredientID
      JOIN ORDER_DETAILS ON HAS_INGREDIENTS.ItemID = ORDER_DETAILS.ItemID
      WHERE ORDER_DETAILS.OrderID = order_id;

      ingredient_name INGREDIENTS.ingredientName%TYPE;
 BEGIN
      OPEN ingredient_cursor;
      LOOP
      FETCH ingredient_cursor INTO ingredient_name;
      EXIT WHEN ingredient_cursor%NOTFOUND;

      -- Existing functionality to display main ingredients
      DBMS_OUTPUT.PUT_LINE('The main ingredient for order ID ' || order_id || ' is
' || ingredient_name);

      -- Insert a new main ingredient (modify as per your table structure)
      INSERT INTO INGREDIENTS (ingredientID, ingredientName)
      VALUES (ingredient_id_sequence.NEXTVAL, 'New Ingredient');

      -- Update an existing main ingredient (modify as per your table structure)
      UPDATE INGREDIENTS
      SET ingredientName = 'Updated Ingredient'
      WHERE ingredientID = 1; -- Modify with an actual ingredient ID



      -- Delete an existing main ingredient (modify as per your table structure)
      DELETE FROM INGREDIENTS
      WHERE ingredientID = 2; -- Modify with an actual ingredient ID
      END LOOP;
      CLOSE ingredient_cursor;
 END;

DECLARE
      order_id ORDERS.OrderID%TYPE := '802';
 BEGIN
      main_ingredients_for_order(order_id);
```

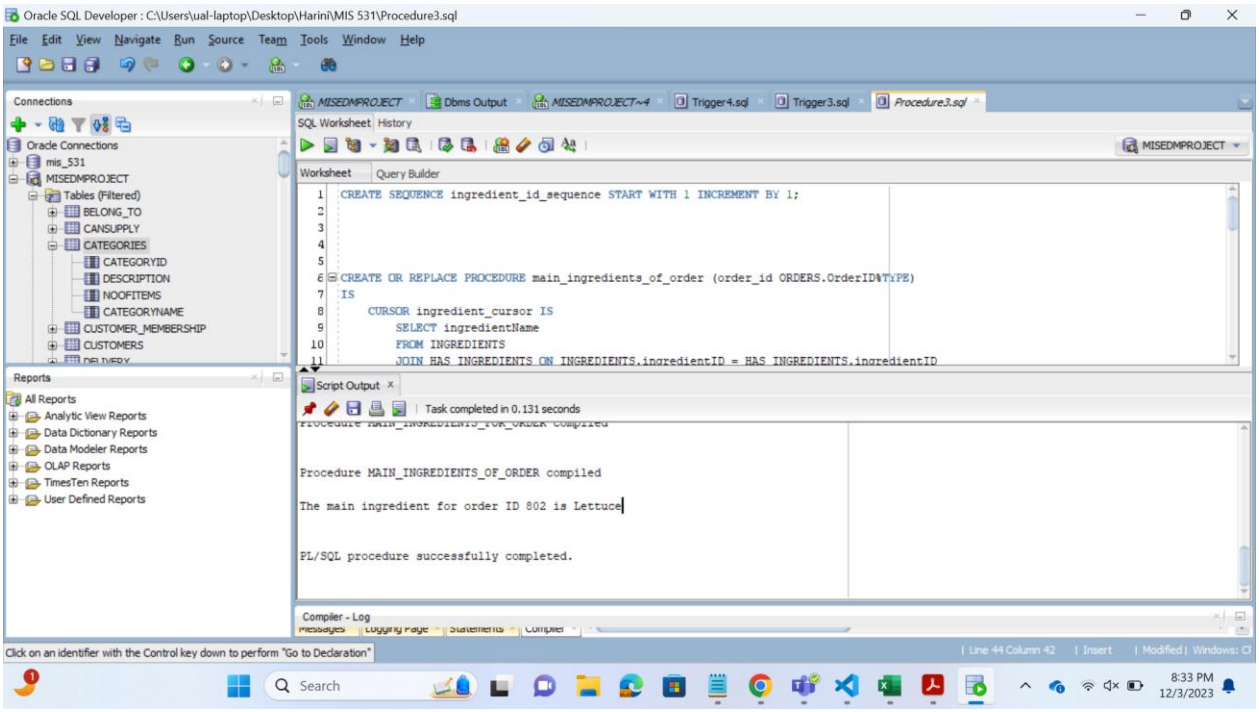
```
END;
/
```

Explanation:

The provided code defines an Oracle PL/SQL stored procedure named main_ingredients_of_order. The purpose of this procedure is to retrieve and display the main ingredients for a specified order, and simultaneously, it demonstrates the insertion of a new main ingredient, updating an existing main ingredient, and deleting an existing main ingredient from the INGREDIENTS table. The procedure utilizes a cursor to fetch main ingredients associated with a given order ID.

The functionality of this procedure is to display the main ingredients associated with a given order ID. Additionally, it showcases the ability to perform various operations on the INGREDIENTS table, such as inserting a new main ingredient, updating an existing main ingredient, and deleting an existing main ingredient. This procedure is helpful for managing and updating main ingredients in the context of orders, providing flexibility for menu adjustments or ingredient modifications based on business needs. It allows for dynamic and real-time changes to the main ingredients associated with specific orders.

Output:



Trigger: Automated Category Item Count Maintenance Trigger

```
CREATE OR REPLACE TRIGGER update_category_items
 AFTER INSERT OR DELETE ON BELONG_TO
 FOR EACH ROW
 BEGIN
      IF INSERTING THEN
      UPDATE CATEGORIES SET noOfItems = noOfItems + 1
      WHERE CategoryID = :NEW.CategoryID;
      ELSIF DELETING THEN
      UPDATE CATEGORIES SET noOfItems = noOfItems - 1
      WHERE CategoryID = :OLD.CategoryID;
      END IF;
 END;

-- Insert a record into BELONG_TO
 INSERT INTO BELONG_TO (CategoryID, ItemID) VALUES (1, 6001);

-- Commit the changes
 COMMIT;

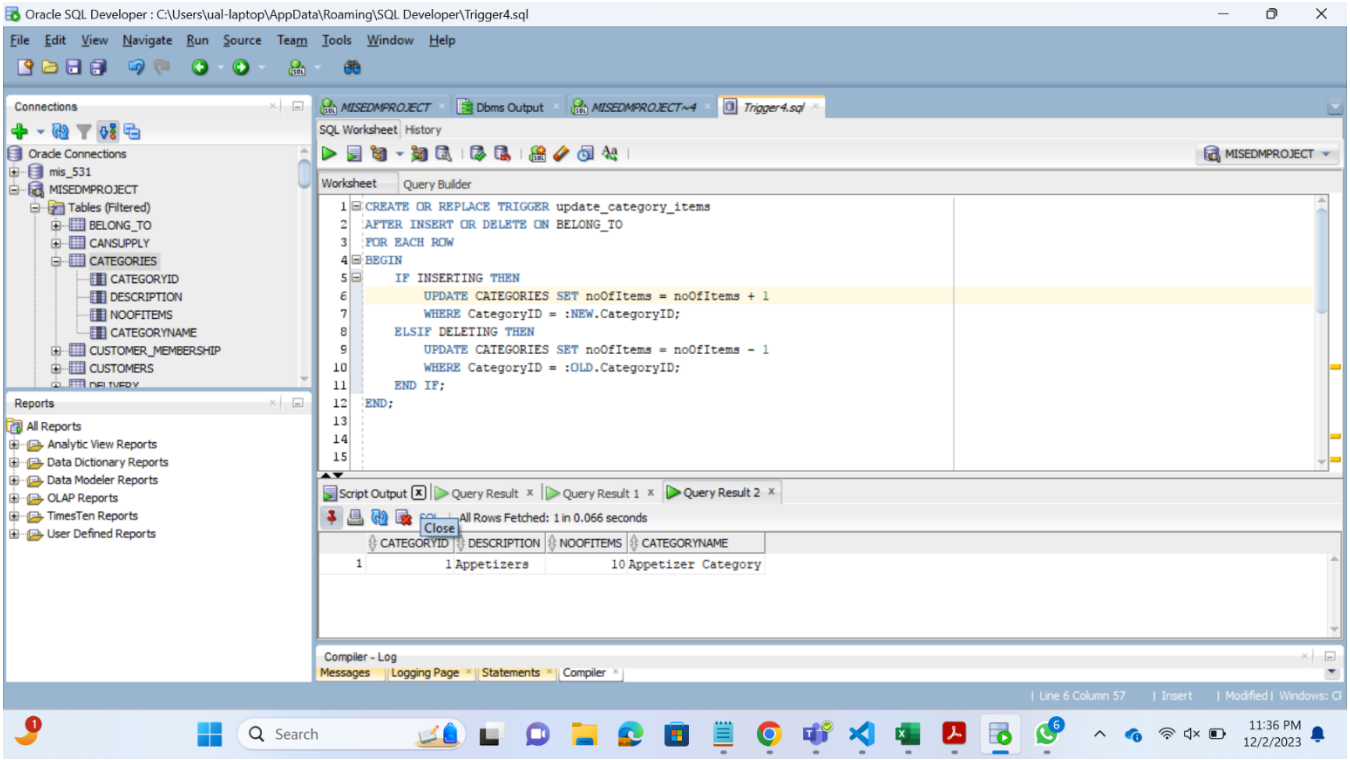
-- Query the updated CATEGORIES table to see the changes
 SELECT * FROM CATEGORIES WHERE CategoryID = 1;

-- Delete a record from BELONG_TO
 DELETE FROM BELONG_TO WHERE CategoryID = 1 AND ItemID = 6001;

-- Commit the changes
 COMMIT;

-- Query the updated CATEGORIES table to see the changes
 SELECT * FROM CATEGORIES WHERE CategoryID = 1;
```

OUTPUT:



Explanation:

The trigger named update_category_items is designed to automatically update the noOfItems column in the CATEGORIES table whenever a new record is inserted into the BELONG_TO table or an existing record is deleted. This trigger operates for each row affected by the insert or delete operation in the BELONG_TO table. If a new record is inserted, it increments the noOfItems count for the corresponding category; if a record is deleted, it decrements the count.

Benefits:

- **Automated Count Maintenance:** The trigger automates the process of updating the noOfItems count in the CATEGORIES table based on changes in the BELONG_TO table. This ensures that the count remains accurate and reflects the current number of items in each category without manual intervention.
- **Consistency and Data Integrity:** By using a trigger, the update operation on the CATEGORIES table is tightly coupled with the insert and delete operations on the BELONG_TO table. This helps maintain consistency and data integrity, preventing discrepancies between the category item count and the actual records in the BELONG_TO table.

Chapter 6:

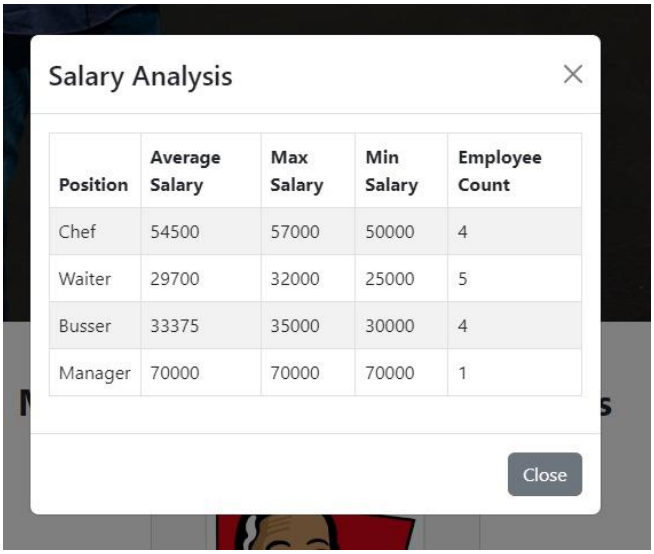
User Interface

Link to our project: http://ec2-35-92-168-0.us-west-2.compute.amazonaws.com:3000/

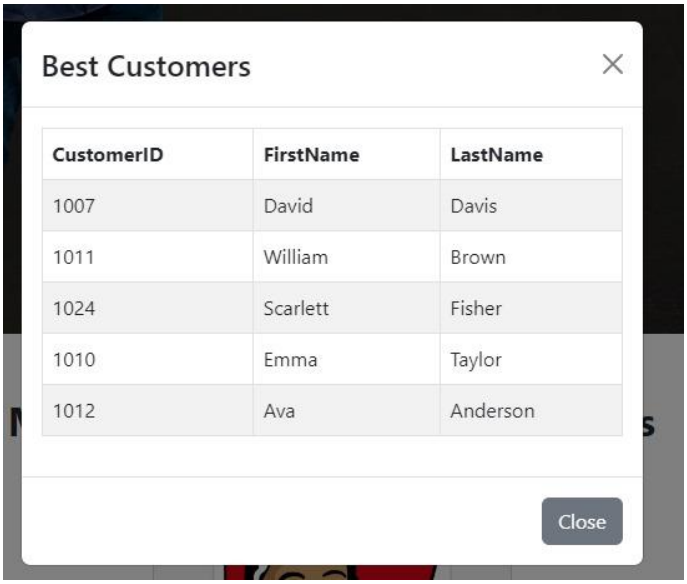
Home page – helps nagivate to different location on the application (after logging in)

Manage various analysis such as salary, customer, and delivery summary from the home page

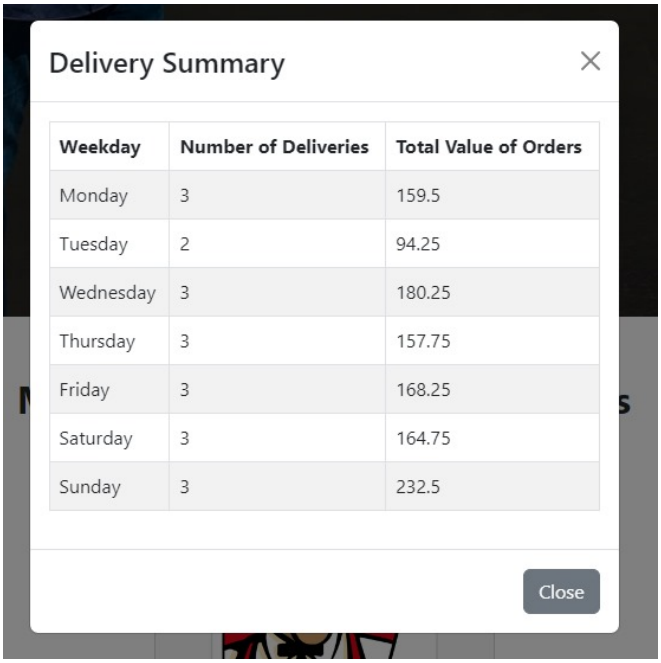
Salary analysis:



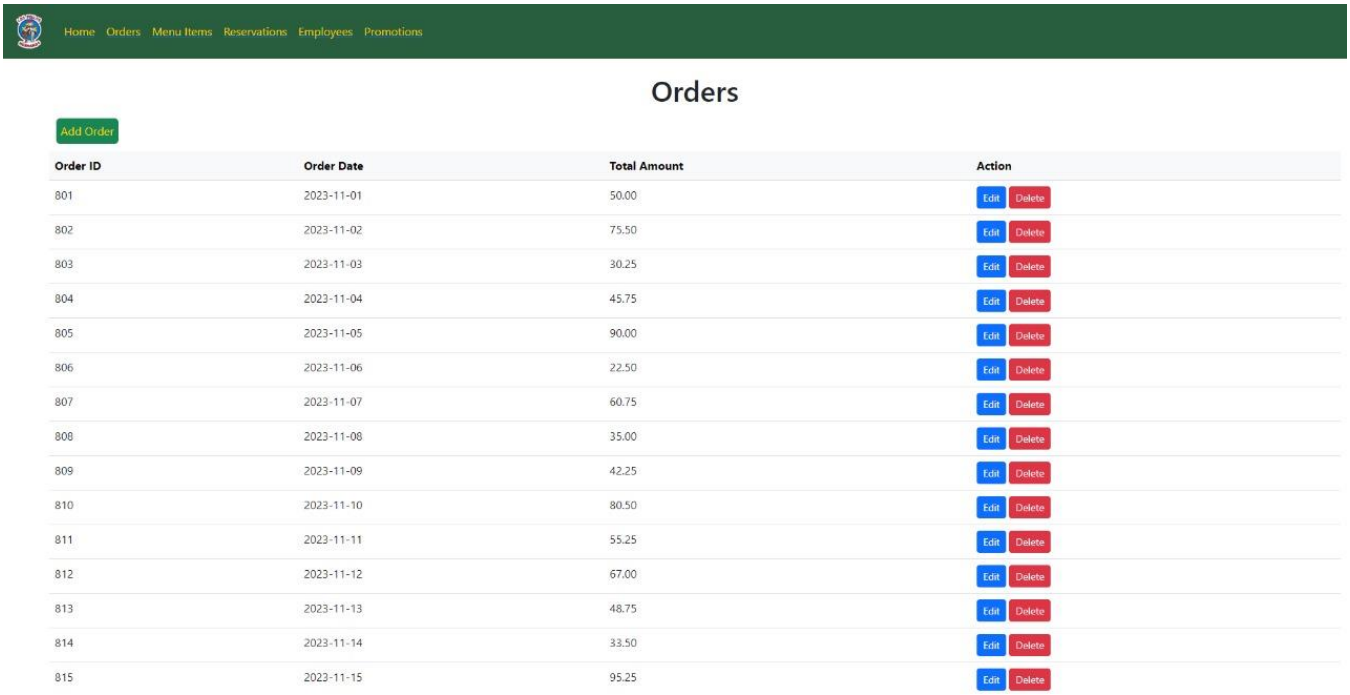
Customer analysis:



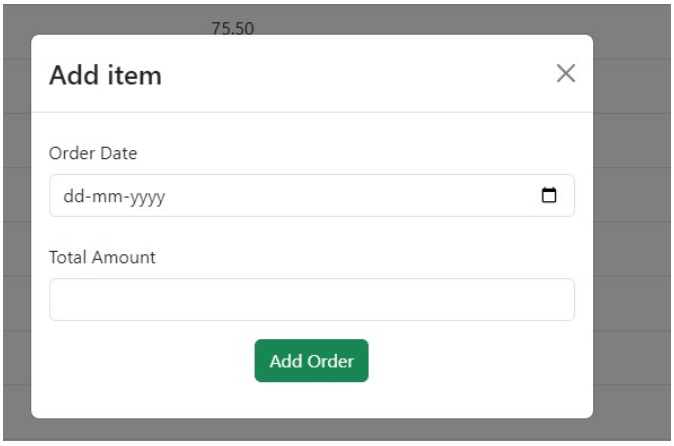
Delivery summary:



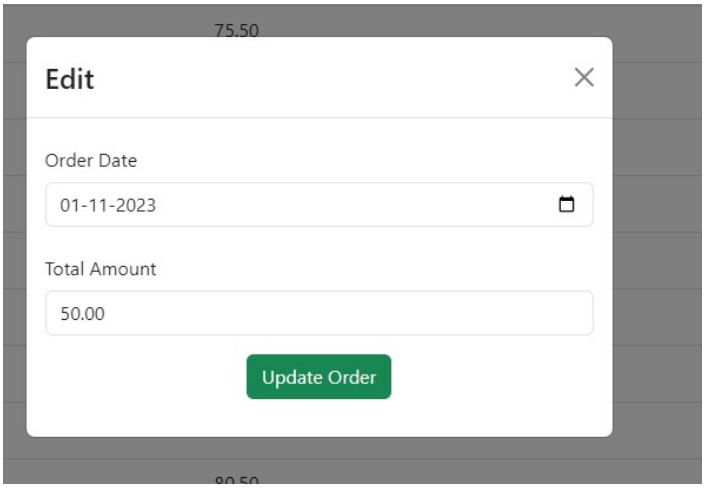
Order page – accessible from the home page. User can view the orders placed to the restaurant



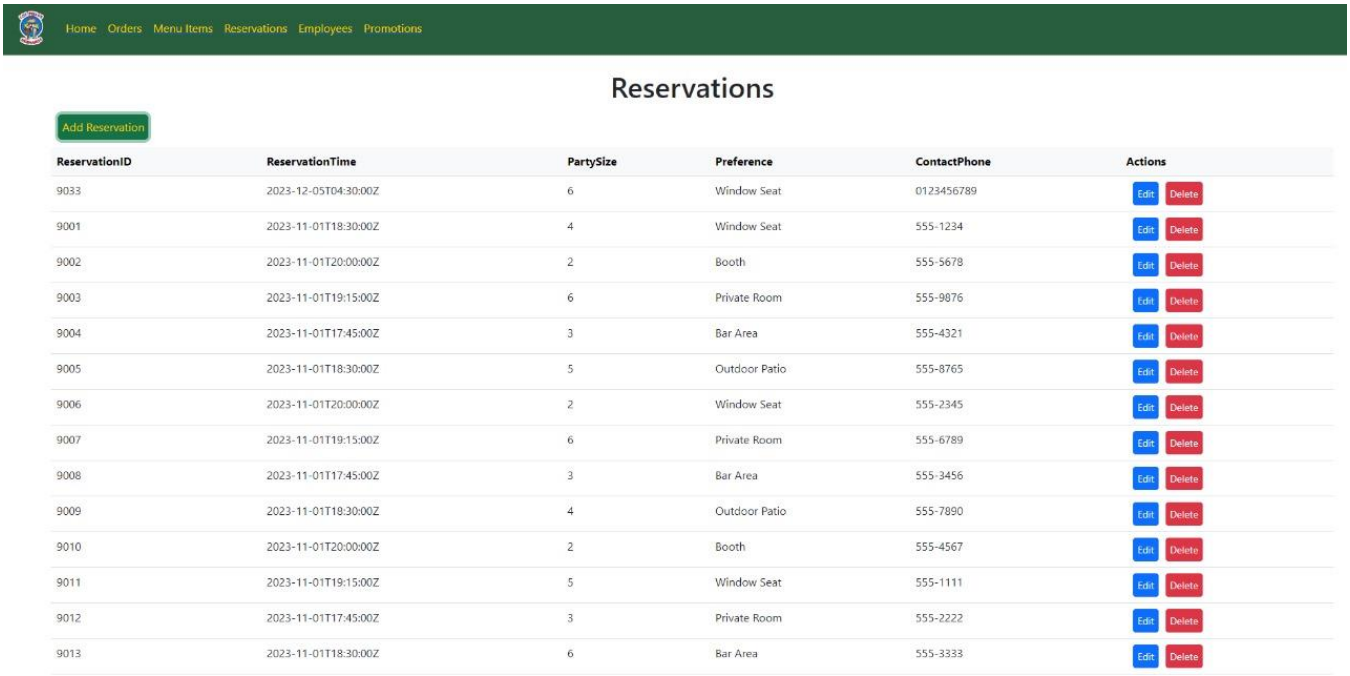
User can add a new order:



User can edit an order:

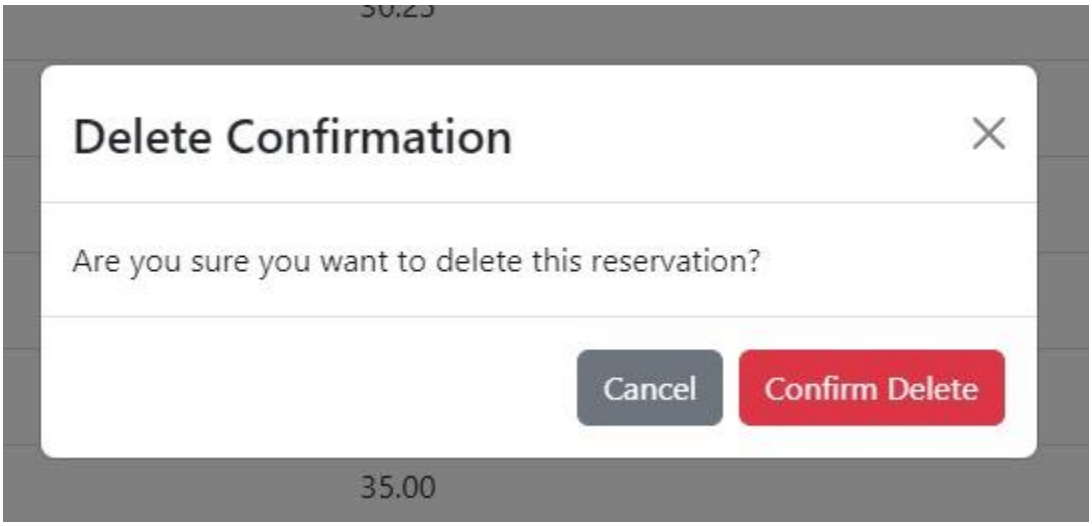


User can access reservations from home page where they can manage their reservations:

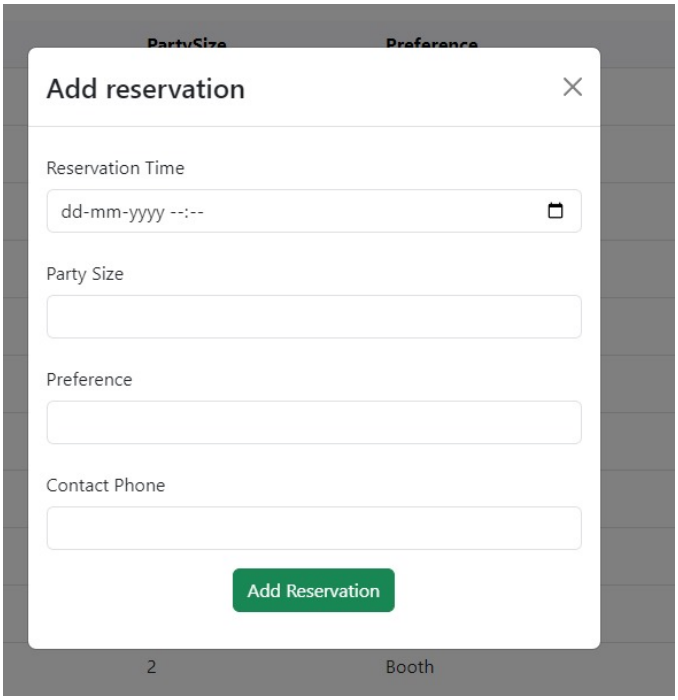


ReservationID	ReservationTime	PartySize	Preference	ContactPhone	Actions
9033	2023-12-05T04:30:00Z	6	Window Seat	0123456789	Edit Delete
9001	2023-11-01T18:30:00Z	4	Window Seat	555-1234	Edit Delete
9002	2023-11-01T20:00:00Z	2	Booth	555-5678	Edit Delete
9003	2023-11-01T19:15:00Z	6	Private Room	555-9876	Edit Delete
9004	2023-11-01T17:45:00Z	3	Bar Area	555-4321	Edit Delete
9005	2023-11-01T18:30:00Z	5	Outdoor Patio	555-8765	Edit Delete
9006	2023-11-01T20:00:00Z	2	Window Seat	555-2345	Edit Delete
9007	2023-11-01T19:15:00Z	6	Private Room	555-6789	Edit Delete
9008	2023-11-01T17:45:00Z	3	Bar Area	555-3456	Edit Delete
9009	2023-11-01T18:30:00Z	4	Outdoor Patio	555-7890	Edit Delete
9010	2023-11-01T20:00:00Z	2	Booth	555-4567	Edit Delete
9011	2023-11-01T19:15:00Z	5	Window Seat	555-1111	Edit Delete
9012	2023-11-01T17:45:00Z	3	Private Room	555-2222	Edit Delete
9013	2023-11-01T18:30:00Z	6	Bar Area	555-3333	Edit Delete

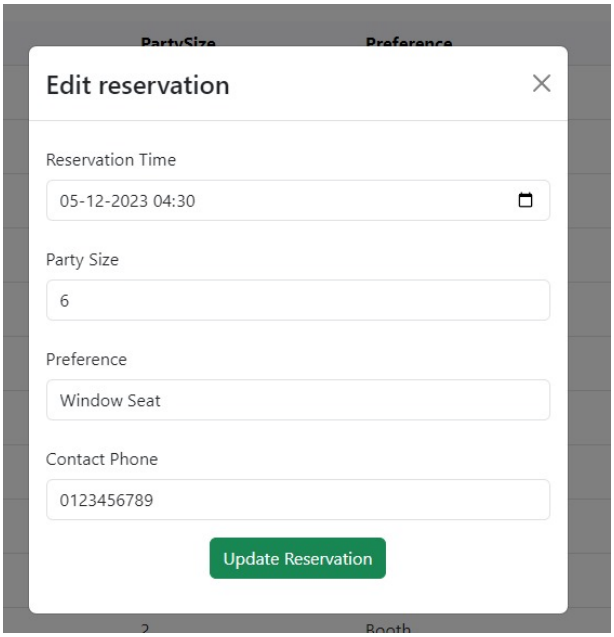
User can delete reservation:



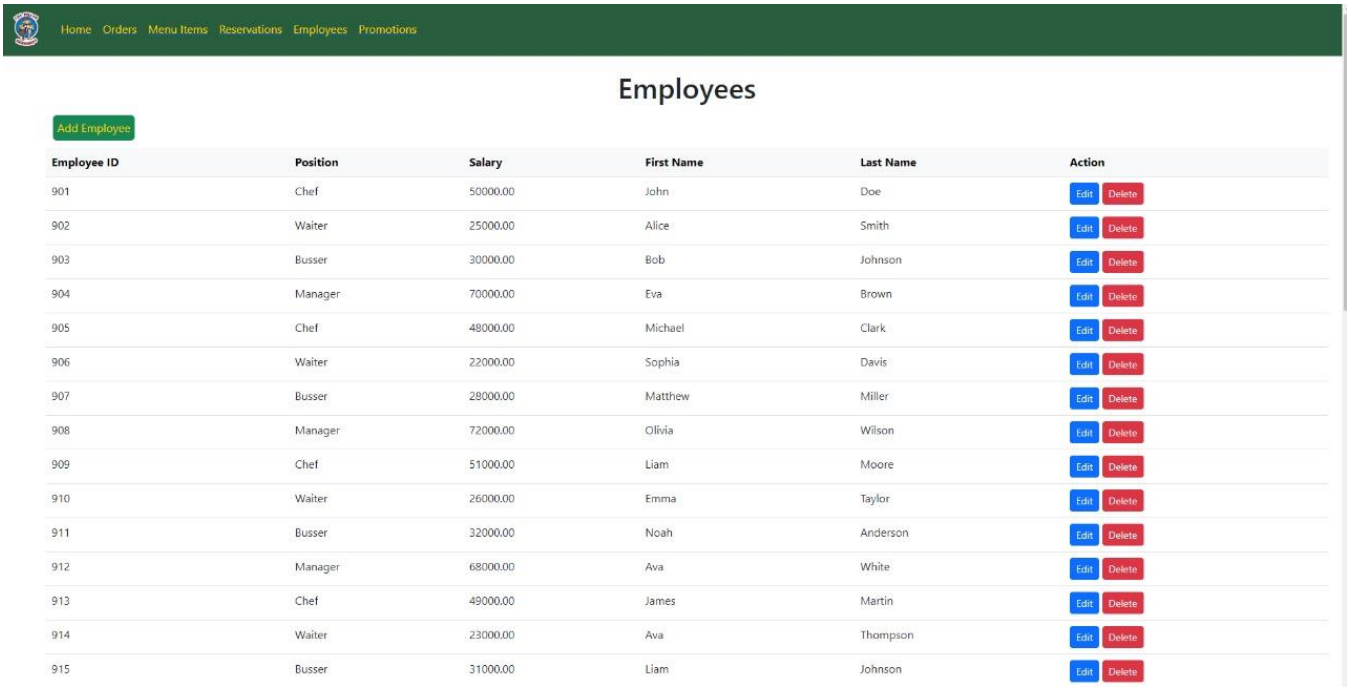
Add reservation:



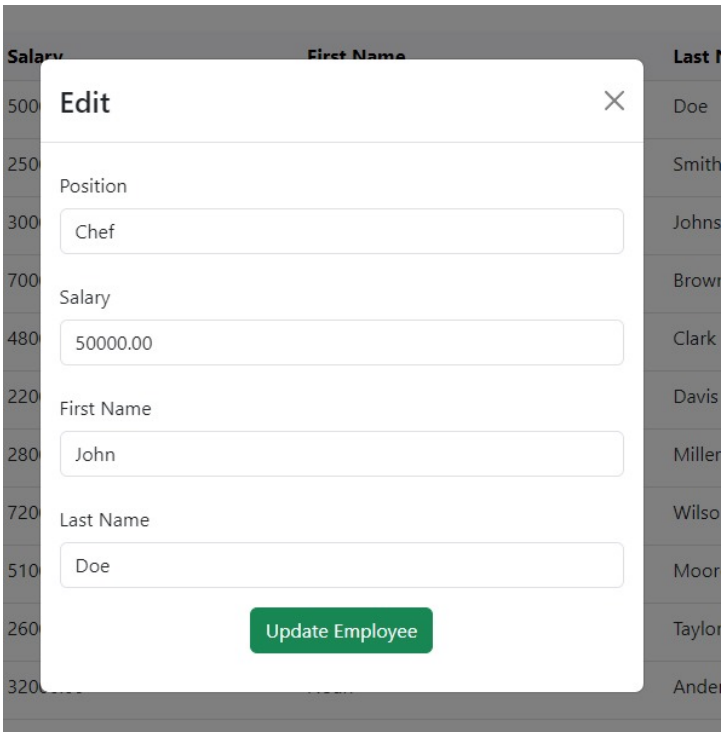
Edit reservation:



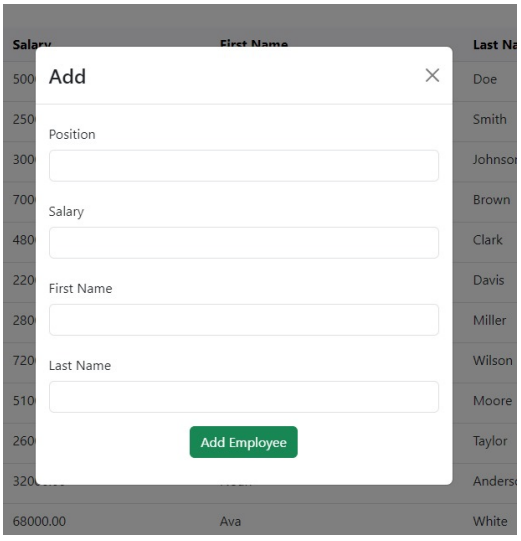
User can manage employees accessed from home page:



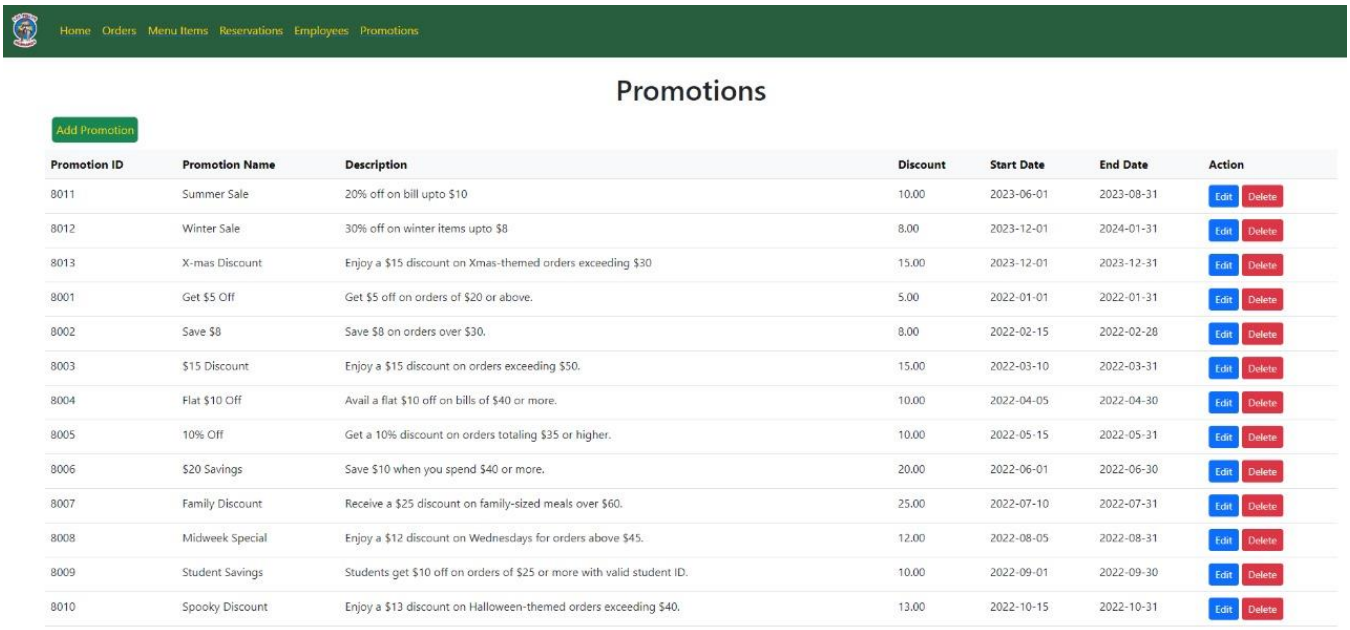
User can edit and delete employees:



User can add employee:

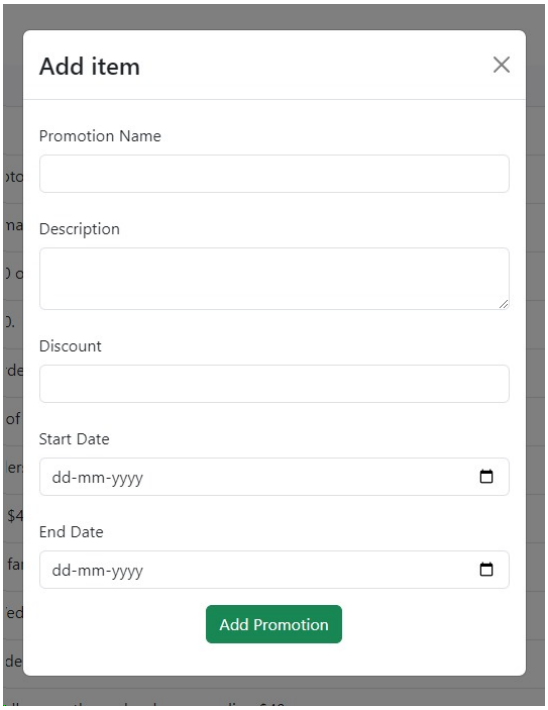


User can manage promotions:

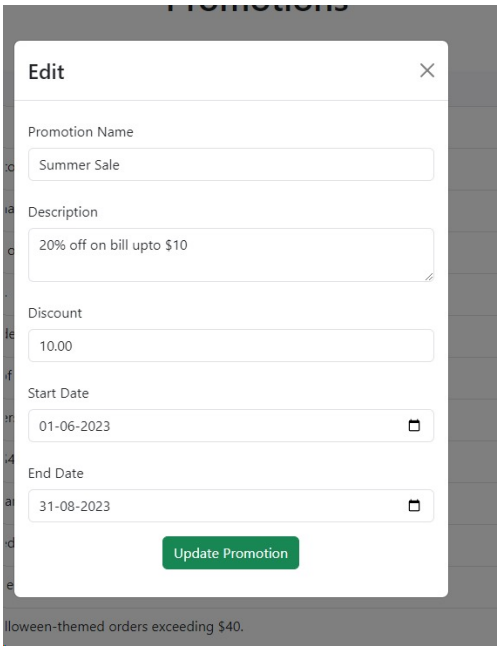


Promotion ID	Promotion Name	Description	Discount	Start Date	End Date	Action
8011	Summer Sale	20% off on bill upto $10	10.00	2023-06-01	2023-08-31	Edit Delete
8012	Winter Sale	30% off on winter items upto $8	8.00	2023-12-01	2024-01-31	Edit Delete
8013	X-mas Discount	Enjoy a $15 discount on Xmas-themed orders exceeding $30	15.00	2023-12-01	2023-12-31	Edit Delete
8001	Get $5 Off	Get $5 off on orders of $20 or above.	5.00	2022-01-01	2022-01-31	Edit Delete
8002	Save $8	Save $8 on orders over $30.	8.00	2022-02-15	2022-02-28	Edit Delete
8003	$15 Discount	Enjoy a $15 discount on orders exceeding $50.	15.00	2022-03-10	2022-03-31	Edit Delete
8004	Flat $10 Off	Avail a flat $10 off on bills of $40 or more.	10.00	2022-04-05	2022-04-30	Edit Delete
8005	10% Off	Get a 10% discount on orders totaling $35 or higher.	10.00	2022-05-15	2022-05-31	Edit Delete
8006	$20 Savings	Save $10 when you spend $40 or more.	20.00	2022-06-01	2022-06-30	Edit Delete
8007	Family Discount	Receive a $25 discount on family-sized meals over $60.	25.00	2022-07-10	2022-07-31	Edit Delete
8008	Midweek Special	Enjoy a $12 discount on Wednesdays for orders above $45.	12.00	2022-08-05	2022-08-31	Edit Delete
8009	Student Savings	Students get $10 off on orders of $25 or more with valid student ID.	10.00	2022-09-01	2022-09-30	Edit Delete
8010	Spooky Discount	Enjoy a $13 discount on Halloween-themed orders exceeding $40.	13.00	2022-10-15	2022-10-31	Edit Delete

User can add the promotion applied:



User can edit the promotion applied:



Chapter 7:

Implementation plan

Project Overview:

Develop a user interface for Los Pollos Hermanos' managers to manage operations internally. The project involves creating a database, developing a user interface, and hosting it on AWS EC2.

Scope:

- Database design and implementation.
- Frontend development for the user interface.
- Integration with APIs.
- Security measures for data protection.
- Testing for functionality and security.
- Deployment on AWS EC2.

Platform:

- Database: Oracle SQL, PostgreSQL, or similar.
- Frontend: React, Angular.
- Backend: Node.js, Python Flask/Django.
- Hosting: AWS EC2.

Steps for Implementation:

1. Database Setup

- Action Items: Create database schema, tables, and relationships based on ER diagram. Populate the database with initial data.
- Person-Hours Estimate: 80-120 hours.
- Cost Estimate:
    - Developer Salary: Approximately $40-60 per hour.

Total Cost: 7,200

- Software Licenses: Open-source database (e.g., Oracle SQL, PostgreSQL).

2. Frontend Development

- Action Items: Design and develop the user interface. Integrate with backend APIs.
- Person-Hours Estimate: 150-200 hours.
- Cost Estimate:
    - Developer Salary: Approximately $40-60 per hour.

3. Security Implementation

- Action Items: Implement authentication(OAuth 2.0), authorization, and data encryption(SHA-256).
- Person-Hours Estimate: 50-80 hours.
- Cost Estimate:
    - Security tools and software: OAuth 2.0 Implementation:
    - Development Costs: $3,000 - $15,000 (integration and customization of OAuth 2.0 in the application)
    - Third-Party Service Costs: Variable; some providers offer free tiers, while premium services can cost $100 - $1,000+ per month.

4. Testing

- Action Items: Functional testing, performance testing, security testing.

- Cost Estimate: Included in developer hours.

5. Deployment

- Action Items: Deploy application on AWS EC2. Set up monitoring and logging.
- Person-Hours Estimate: 30-50 hours.
- Cost Estimate:
    - AWS EC2 estimated costs:
        - Monthly: $6000.00
        - Yearly: $72000.00
    - Cost is estimated by 20 EC2 instances and a storage of 1000 GB
    - Total Person-Hours: 310-450 hours.

Summary of Cost Breakdown:

- Developer Salary: $12,400 - $27,000 (based on $40-60/hour).
- Software Licenses: Minimal (assuming open-source tools).
- Security Tools: $3,000 - $15,000 (integration and customization of OAuth 2.0 in the application).
    - AWS EC2 Hosting: Varies based on usage (use AWS Pricing Calculator for estimates).

Appendix A: Lessons Learned

Importance of thorough planning in database design to avoid future scalability issues.

The necessity of incorporating security measures from the beginning.

The value of iterative testing throughout development.

The benefits of clear communication and regular updates within the team.

Learning from other groups: The effectiveness of different project management methodologies, the impact of technology choices on development speed and maintenance, and the significance of user experience in interface design.

References:

AWS Pricing Calculator: AWS Pricing

Developer Salary Estimates: Glassdoor

Open Source Database Information: Oracle SQL, PostgreSQL