CIST 3222 DATABASE SYSTEMS

GROUP PROJECT REPORT

Busy Bean

Group 2

Keil Barracliffe, Linda Mei, Freidalyz Mojica, Jacob Parratto

Project Overview

1. Executive Summary:

Our business, Busy Bean, was founded on the idea that all have a right to quick, cheap, and nutritious refreshments to accompany their busy lives. We specialize in coffee stimulants, with our environmentally-conscious sourced coffee beans. While we have successfully attracted our targeted demographic of students and business workers, we still currently suffer from some financial hurdles. We often find our inventory to be under or over stocked. This issue along with our manual entry speed has limited our ability to properly expand our business.

In order to rectify the financial restrictions and expand the profitability of our business, it was decided that a database must be created. This database will automatically update inventory counts, create and store order receipts, store employee contact information and store employee working hours. This will allow our business to pay the employees the correct amount, expedite order processing, and identify order trends to follow the current market while maintaining an efficient inventory.

Since implementing this database, our company has been able to track the employees hours more efficiently, which makes scheduling much easier. We have also been more prepared with inventory on hand, since the database allows inventory tracking and forecasting. The budget concerns are no longer relevant, since tracking income and expenses is much easier. This increased profit will allow our business to expand to other areas and also better serve our clientele.

2. Who did What:

Everyone: Completed the Business Description. Created and updated the E-R diagram. Completed 2-3 entities' cardinalities and relationship sentences each in the Conceptual Data Model section. Transformed the conceptual model into the relationship model. Completed 2-3 assigned database design summary sentences each in the Database Design section. Completed the meeting minutes in a rotation order. Checked each member's individual work and SQL codes.

Keil Barracliffe: Consistently maintained meeting time limits by suggesting delegation of tasks. Created list of functional dependencies. Wrote SQL statements to create and populate the EMPLOYEE table. Created slides 3 and 4 for final presentation. Troubleshot syntax errors by checking the textbook along with various help sites when our group ran into issues with SQL developer. Wrote the second paragraph and the rest of the Executive Summary.

Linda Mei: Created the group project report and kept the report up-to-date and well-formatted. Kept the team well-organized and structured for meetings, deadlines, and tasks. Completed 2, 3, 6, and 7 in the Business Description section. Used SQL codes to create INGREDIENT and INGREDIENT_USE tables with constraints. Used SQL codes to insert data into the two tables. Tested all of the tables by creating and inserting data into the tables. Created slides 1, 2, and the Data Implementation slides.

Freidalyz Mojica: Created slides 5 and 6 for final presentation. Used SQL statements to create the EMPLOYEE_HOURS and ITEM table. Similarly, used SQL INSERT statements to populate the EMPLOYEE_HOURS and ITEM tables as well. Often shared documents/tables with the professor in order to get feedback. Also, validated and implemented all of the group's SQL statements, in order to create the Busy Bean database and the relational diagram in SQL Developer.

Jacob Parratto: Created example data in Excel to create tables and also helped come up with the entities that our database would consist of. Tested all of the tables by creating the tables and then inserting data into the tables. Used SQL codes to create ORDERS and ORDER_ITEM tables with constraints. Created the slides detailing the relations and cardinality between the tables. Wrote the introduction paragraph for the Executive Summary.

3. Table of Contents:

Business Description…………………………………………………………………5

Name of the Business……………………………………………………………...5

Purpose of the Business…………………………………………………………...5

Summary of Business Activities…………………………………………………..5

Problems, Opportunities, and Objectives…………………………………………5

Business Case……………………………………………………………………...5

Information and Data Requirement………………………………………………6

List of Entities……………………………………………………………………..6

Conceptual Data Model…………………………………………………………...7

Entity Relationship Diagram………………………………………………………7

Database Design ………………………………………………………………….9

E-R Model into Relationship Model………………………………………………9

List of Functional Dependencies…………………………………………………...11

Database Implementation……………………………………………………….....11

SQL Codes for Normalized Table……………………………………………….....11

SQL Codes for Inserting Data……………………………………………………15

ER Diagram in SQL Developer…………………………………………………25

Tables in SQL Developer………………………………………………………....26

Business Description

1. **Name of the Business**: Busy Bean.

2. **Purpose of the Business**: A coffeehouse which provides refreshments for its customers. The purpose of these refreshments is to stimulate the consumers' cognitive functions and give them a boost of energy.

3. **Summary of Business Activities**:
   Busy Bean's objective is to advocate for environmental and mental health awareness through the use of ethical, sustainable, and high quality sourced coffee beans. The targeted customer base of the business are students and business workers.

4. **Problems, Opportunities, and Objectives:**

   Problems: The problems within the current database is that the system is unable to track the quantity of ingredients which results in over-ordering and under-ordering stock for the business. Furthermore, originally the business rang customers up manually and did not have a database that was suitable for the business' increase in customers.

   Opportunities: The business can employ more workers, put out more products for sale, manage overall business funds better, and optimize time and sales.

   Objectives: The new database system will manage business funds better, optimize time and sales, and predict supply and demand. In addition, the database will allow the business to maximize the efficiency of service and provide an opportunity for business growth.

5. **Business Case (i.e., reasons or justifications for the new database system)**: To ensure that the database meets the demands of the business to manage the business better

overall. With the creation of a database system, the business can transform from a small business and increase in dividends. Furthermore, the business can expand the menu and integrate new ingredients for products that keep up with the latest trends.

6. **Information and Data Requirement**:

The data that will be required is the "Item ID," "Item Name," "Price," "OrderNumber," "Item Quantity," "Employee ID," "Date Worked," "Hours Worked," and "Hourly Wage."

The information that will be generated will include: "New Order Item," "Total Labor," "Best Seller," "Price Low to High," and "Price High to Low."

The data obtained from Item ID, Order Number, Item Name, Price, and Item Quantity is necessary to determine "New Order Item." The data from Employee ID, Date Worked, Hours Worked, and Hourly Wage would generate information for "Total Labor." The information generated for "Best Seller" is determined by OrderNumber, Item ID, Item Name, Price, and Item Quantity. The data from Item ID, Item Name, and Price would determine "Price Low to High" and "Price High to Low."

7. **List of Entities (tables) that have been identified**:

EMPLOYEE(EmployeeID, FirstName, LastName, Address, Phone, Email, HourlyWage)

EMPLOYEE_HOURS(EmployeeID, DateWorked, HoursWorked)

ORDERS(OrderNumber, EmployeeID, OrderDate, OrderSubtotal, TotalTax, Total)

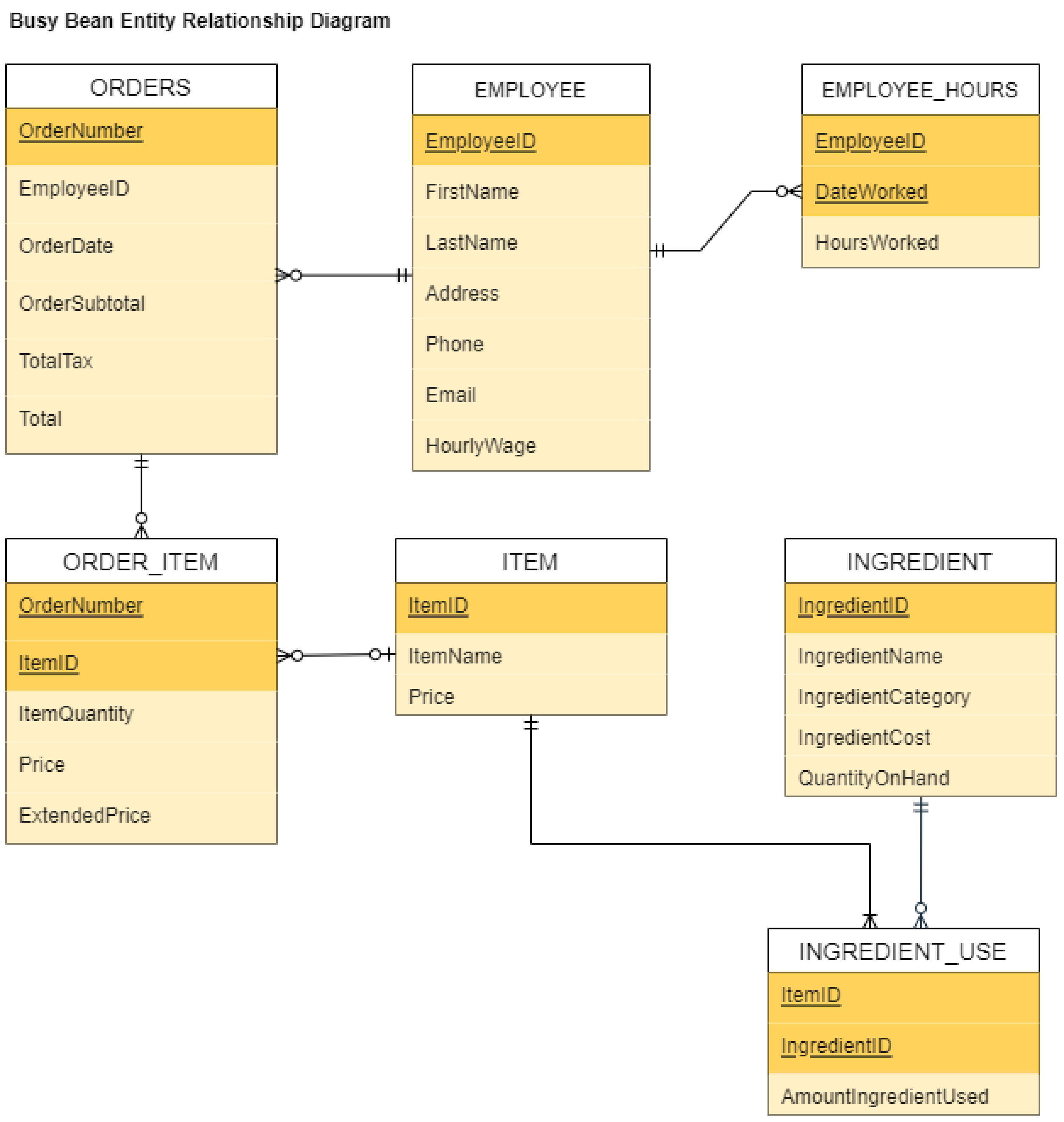
ORDER_ITEM(OrderNumber, ItemID, ItemQuantity, Price, ExtendedPrice)

ITEM(ItemID, ItemName, Price)

INGREDIENT(IngredientID, IngredientName, IngredientCategory, IngredientCost, QuantityOnHand)

INGREDIENT_USE(ItemID, IngredientID, AmountIngredientUsed)

Conceptual Data Model



The conceptual data model contains seven entities which form six different relationships between the entities.

For the ITEM entity to INGREDIENT_USE entity, there is a maximum cardinality of one-to-many and a minimum cardinality of mandatory to mandatory. An item must contain and use at least one ingredient, but can also use many ingredients. There must be an ingredient used in each item.

For the INGREDIENT entity to INGREDIENT_USE entity, there is a maximum cardinality of one-to-many and a minimum cardinality of mandatory to optional. An ingredient can be used for an item zero, one, or many times. An instance of an ingredient being used for that item must belong to at least one ingredient, and therefore will only belong to that one ingredient.

For the EMPLOYEE entity to EMPLOYEE_HOURS entity, there is a maximum cardinality of one-to-many and a minimum cardinality of mandatory to optional. An employee can have zero, one, or many employee hours that belong to them. Employee hours must belong to one and only one employee who works those hours.

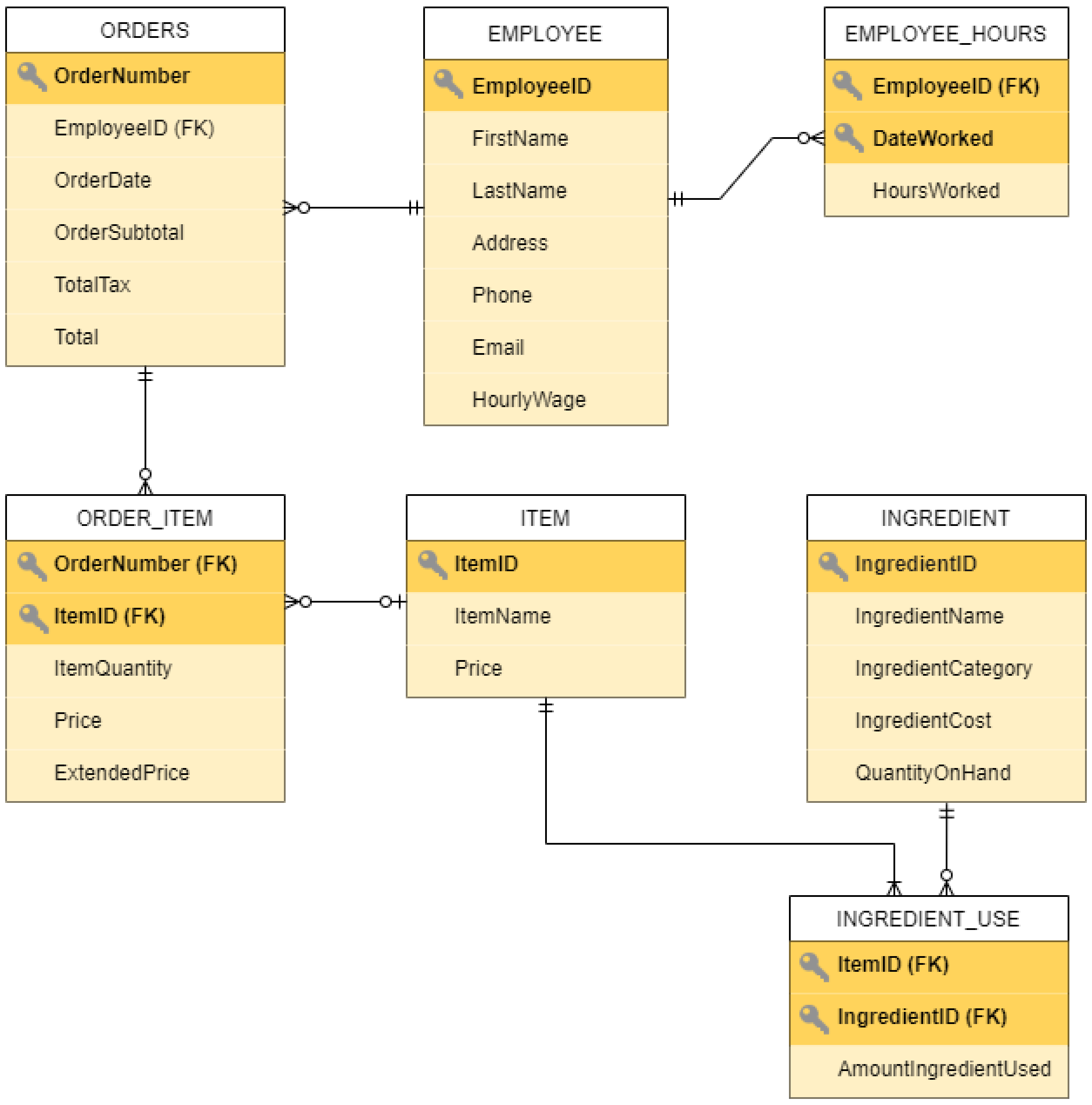
For the ITEM entity to ORDER_ITEM entity, there is a maximum cardinality of one-to-many and a minimum cardinality of optional to optional. An item can belong to zero, one, or many order items (order items are the instances in which a customer's order contains an item). An order item can belong to one item or no item on the menu.

For the EMPLOYEE entity to the ORDERS entity, there is a maximum cardinality of one-to-many and a minimum cardinality of mandatory to optional. One order must be associated with at most one employee. However, one employee can complete zero orders if they did not work that day or they could complete one or many orders by working that day.

For the ORDERS entity to the ORDER_ITEM entity, there is a maximum cardinality of one-to-many and a minimum cardinality of mandatory to optional. An order can have zero, one, or many order items. Whereas, an order item must be associated with one and only one order.

Database Design

Busy Bean E-R Model into Relationship Model



The logical and physical model of the database displays six different relationships between the seven entities. The identifiers from the list of entities will become the primary keys and the entities will become the tables.

From the ITEM table to the INGREDIENT_USE table, the parent table is the ITEM table with the primary key called ItemID. The child table is the INGREDIENT_USE table with the composite primary key ItemID and IngredientID. The child table also has two foreign keys: ItemID and IngredientID.

From the INGREDIENT table to the INGREDIENT_USE table, the parent table is the INGREDIENT table with the primary key called IngredientID. The child table is the INGREDIENT_USE table with the composite primary key ItemID and IngredientID. The child table also has two foreign keys: ItemID and IngredientID.

From the EMPLOYEE table to the EMPLOYEE_HOURS table, the parent table is the EMPLOYEE table with the primary key called EmployeeID. The child table is the EMPLOYEE_HOURS table with the composite primary key EmployeeID and DateWorked. The child table has a foreign key called EmployeeID.

From the ITEM table to the ORDER_ITEM table, the parent table is the ITEM table with the primary key called ItemID. The child table is the ORDER_ITEM table with the primary key OrderNumber and ItemID. The child table also has a foreign key called ItemID and another foreign key called OrderNumber.

From the EMPLOYEE table to the ORDERS table, the parent table is the EMPLOYEE table with the primary key EmployeeID. The child table is the ORDERS table, with the primary key OrderNumber, and the foreign key EmployeeID.

From the ORDERS table to the ORDER_ITEM table, the parent table is the ORDERS table with the primary key called OrderNumber. The child table is the ORDER_ITEM table with the primary key OrderNumber and ItemID. The child table also has a foreign key called OrderNumber and another foreign key called ItemID.

2. List of Functional Dependencies:

Functional Dependency of the Employee Table:

EmployeeID   (FirstName, LastName, Address, Phone, Email, HourlyWage)

Functional Dependency of the Employee_Hours Table:

(EmployeeID, DateWorked)   (HoursWorked)

Functional Dependency of the Orders Table:

OrderNumber   (EmployeeID, OrderDate, OrderSubtotal, TotalTax, Total)

Functional Dependency of the Order_Item Table:

(OrderNumber, ItemID)   (ItemQuantity, Price, ExtendedPrice)

Functional Dependency of the Item Table:

ItemID   (ItemName, Price)

Functional Dependency of the Ingredient Table:

IngredientID   (IngredientName, IngredientCategory, IngredientCost, QuantityOnHand)

Functional Dependency of the Ingredient_Use:

(ItemID, IngredientID)   (AmountIngredientUsed)

Database Implementation

1. SQL Codes for creating all Tables

```
CREATE TABLE EMPLOYEE (

EmployeeID          Int                 NOT NULL,

FirstName           Char(35)            NOT NULL,

LastName            Varchar(35)         NOT NULL,

Address             Varchar(45)         NOT NULL,
Phone               Varchar(30)         NOT NULL,
Email               Varchar(50)         NULL,
HourlyWage          Numeric(5,2)        NOT NULL,
CONSTRAINT    EmployeePK      PRIMARY KEY (EmployeeID),
CONSTRAINT    EmployeeAK1     UNIQUE (LastName, FirstName)
);

CREATE TABLE EMPLOYEE_HOURS(
EmployeeID          Int     NOT NULL,
DateWorked          Date    NOT NULL,
HoursWorked         Int     NOT NULL,
CONSTRAINT          E_HIntPK      PRIMARY KEY(EmployeeID, DateWorked),
CONSTRAINT          E_HIntFK      FOREIGN KEY(EmployeeID)
        REFERENCES EMPLOYEE(EmployeeID)
        ON DELETE CASCADE
);

CREATE TABLE ORDERS (
    OrderNumber         Int          NOT NULL,
    EmployeeID          Int          NOT NULL,
    OrderDate           Date         NOT NULL,
    OrderSubtotal       Numeric(5,2)      NOT NULL,
    TotalTax            Numeric(5,2)      NOT NULL,
    Total               Numeric(5,2)       NOT NULL,
    CONSTRAINT          OrderPK          PRIMARY KEY(OrderNumber),
    CONSTRAINT          OrderFK          FOREIGN KEY(EmployeeID)
                        REFERENCES EMPLOYEE(EmployeeID),
    CONSTRAINT     ValidTotal  CHECK ((OrderSubtotal + TotalTax = Total) AND (Total <
1000.00) AND (Total >= 0.00))
    );

CREATE TABLE ITEM (
ItemID                  Int               NOT NULL,
ItemName               Char(45)         NOT NULL,
Price                  Numeric(5,2)            NOT NULL,
CONSTRAINT         ItemPK          PRIMARY KEY(ItemID)
);

CREATE TABLE ORDER_ITEM (
    OrderNumber           Int           NOT NULL,
    ItemID               Int             NOT NULL,
    ItemQuantity           Int             NOT NULL,
    Price               Numeric(5,2)       NOT NULL,
    ExtendedPrice        Numeric(5,2)       NOT NULL,
    CONSTRAINT          Order_ItemPK    PRIMARY KEY(OrderNumber, ItemID),
    CONSTRAINT          OrderFK1       FOREIGN KEY(OrderNumber)
```

```
                        REFERENCES ORDERS(OrderNumber),
    CONSTRAINT          OrderFK2      FOREIGN KEY(ItemID)
                        REFERENCES ITEM(ItemID),
    CONSTRAINT    ValidQuantity  CHECK ( ItemQuantity > 0)
    );

CREATE TABLE INGREDIENT(
IngredientID          Int          NOT NULL,
IngredientName          Char (35)          NOT NULL,
IngredientCategory          Char (35)          NULL,
IngredientCost          Numeric(3, 2)          NOT NULL,
QuantityOnHand          Int          NOT NULL,
CONSTRAINT          IngredientPK          PRIMARY KEY(IngredientID),
CONSTRAINT          IngCategoryConstraint          CHECK(IngredientCategory IN ('Coffee
Beans', 'Dairy', 'Coffee Creamers', 'Other')),
CONSTRAINT          IngCostConstraint          CHECK((IngredientCost >= 0.01) AND
(IngredientCost <= 0.60)),
CONSTRAINT          QOHConstraint          CHECK(QuantityOnHand >= 0)
);

CREATE TABLE INGREDIENT_USE(
ItemID          Int          NOT NULL,
IngredientID          Int          NOT NULL,
AmountIngredientUsed        Int        NULL,
CONSTRAINT        Ingredient_UsePK        PRIMARY KEY(ItemID, IngredientID),
CONSTRAINT        ItemFK1        FOREIGN KEY(ItemID)
                        REFERENCES ITEM(ItemID),
CONSTRAINT        ItemFK2        FOREIGN KEY(IngredientID)
                        REFERENCES INGREDIENT(IngredientID),
CONSTRAINT        AIUConstraint        CHECK(AmountIngredientUsed >= 0)
);
```

2. SQL Codes for Inserting Data

```
/*        INSERT INTO EMPLOYEE TABLE        */
CREATE SEQUENCE seqEID Increment by 1 start with 1;
        INSERT INTO EMPLOYEE
        ( EmployeeID, FirstName, LastName, Address, Phone, Email, HourlyWage)
        VALUES(seqEID.NextVal, 'Joe', 'Smith', '123 Oak Street Hammonton NJ 08037',
        '856-834-5000', 'jsmith5000@gmail.com', 15.00);

        INSERT INTO EMPLOYEE
        ( EmployeeID, FirstName, LastName, Address, Phone, Email, HourlyWage)
        VALUES(seqEID.NextVal, 'Keil', 'Barracliffe', '400 Newton Ave Oaklyn NJ 08107',
        '856-783-2666', 'keil.barracliffe@gmail.com', 15.00);

        INSERT INTO EMPLOYEE
        ( EmployeeID, FirstName, LastName, Address, Phone, Email, HourlyWage)
        VALUES(seqEID.NextVal, 'Linda', 'Mei', '1000 Beach Street Atlantic City NJ 08201',
        '609-434-5000', 'linda_mei@gmail.com', 17.50);

        INSERT INTO EMPLOYEE
        ( EmployeeID, FirstName, LastName, Address, Phone, Email, HourlyWage)
        VALUES(seqEID.NextVal, 'Freidalyz', 'Mojica', '999 Boardwalk Ave Atlantic City NJ
        08201', '609-434-1800', 'mojica.freida@gmail.com', 16.75);

        INSERT INTO EMPLOYEE
        ( EmployeeID, FirstName, LastName, Address, Phone, Email, HourlyWage)
        VALUES(seqEID.NextVal, 'Jacob', 'Parratto', '500 Brown Rd Egg Harbor Twp NJ 08234',
        '856-939-9494', 'jp@gmail.com', 19.00);

/*      INSERT INTO EMPLOYEE_HOURS TABLE        */
INSERT INTO EMPLOYEE_HOURS
        (EmployeeID, DateWorked, HoursWorked)
        VALUES(1, TO_DATE('04/25/2022', 'MM/DD/YYYY'), 24);

INSERT INTO EMPLOYEE_HOURS
        (EmployeeID, DateWorked, HoursWorked)
        VALUES(2, TO_DATE('04/26/2022', 'MM/DD/YYYY'), 32);
```

```
INSERT INTO EMPLOYEE_HOURS
        (EmployeeID, DateWorked, HoursWorked)
        VALUES(3, TO_DATE('04/27/2022', 'MM/DD/YYYY'), 40);

INSERT INTO EMPLOYEE_HOURS
        (EmployeeID, DateWorked, HoursWorked)
        VALUES(4, TO_DATE('04/29/2022', 'MM/DD/YYYY'), 18);

INSERT INTO EMPLOYEE_HOURS
        (EmployeeID, DateWorked, HoursWorked)
        VALUES(5, TO_DATE('04/29/2022', 'MM/DD/YYYY'), 40);
```

/*   INSERT data for ORDERS   */

```
Create Sequence seqOrderNum Increment by 1 start with 1;
INSERT INTO ORDERS
        (OrderNumber, EmployeeID, OrderDate, OrderSubtotal, TotalTax,
         Total)
        VALUES (
seqOrderNum.NextVal, 1, '10-APRIL-2022', 9.50, 0.67, 10.17);

INSERT INTO ORDERS
        (OrderNumber, EmployeeID, OrderDate, OrderSubtotal, TotalTax,
         Total)
        VALUES (
        seqOrderNum.NextVal, 2, '11-APRIL-2022', 3.00, 0.21, 3.21);

INSERT INTO ORDERS
        (OrderNumber, EmployeeID, OrderDate, OrderSubtotal, TotalTax,
         Total)
        VALUES (
        seqOrderNum.NextVal, 3, '12-APRIL-2022', 5.50, 0.39, 5.89);

INSERT INTO ORDERS
        (OrderNumber, EmployeeID, OrderDate, OrderSubtotal, TotalTax,
         Total)
        VALUES (
        seqOrderNum.NextVal, 4, '13-APRIL-2022', 10.75, 0.75, 11.50);

INSERT INTO ORDERS
        (OrderNumber, EmployeeID, OrderDate, OrderSubtotal, TotalTax,
         Total)
        VALUES (
        seqOrderNum.NextVal, 5, '14-APRIL-2022', 6.00, 0.42, 6.42);


/*      INSERT data for ITEM        */
```

```
CREATE SEQUENCE seqIID Increment by 1 start with 1;


INSERT INTO ITEM
        (ItemID, ItemName, Price)
        VALUES(seqIID.NextVal, 'Black Coffee','3.00');
INSERT INTO ITEM
        (ItemID, ItemName, Price)
        VALUES(seqIID.NextVal, 'Iced Coffee', '3.75');
INSERT INTO ITEM
        (ItemID, ItemName, Price)
        VALUES(seqIID.NextVal, 'Espresso', '2.50');
INSERT INTO ITEM
        (ItemID, ItemName, Price)
        VALUES(seqIID.NextVal, 'Steamed Milk', '1.75');
INSERT INTO ITEM
        (ItemID, ItemName, Price)
        VALUES(seqIID.NextVal, 'Bagels', '1.50');
INSERT INTO ITEM
        (ItemID, ItemName, Price)
        VALUES(seqIID.NextVal, 'Scones', '2.00');
INSERT INTO ITEM
        (ItemID, ItemName, Price)
        VALUES(seqIID.NextVal, 'Cookies', '3.00');
```

```
/*          INSERT data for ORDER_ITEM          */
INSERT INTO ORDER_ITEM
        (OrderNumber, ItemID, Price, ItemQuantity,
         ExtendedPrice)
        VALUES (
        00001, 001, 3.00, 2, 6.00);

INSERT INTO ORDER_ITEM
        (OrderNumber, ItemID, Price, ItemQuantity,
         ExtendedPrice)
        VALUES (
        00001, 004, 1.75, 2, 3.50);

INSERT INTO ORDER_ITEM
        (OrderNumber, ItemID, Price, ItemQuantity,
         ExtendedPrice)
        VALUES (
        00002, 001, 3.00, 1, 3.00);

INSERT INTO ORDER_ITEM
        (OrderNumber, ItemID, Price, ItemQuantity,
         ExtendedPrice)
```

```
          VALUES (
          00003, 006, 2.00, 1, 2.00);

INSERT INTO ORDER_ITEM
          (OrderNumber, ItemID, Price, ItemQuantity,
           ExtendedPrice)
          VALUES (
          00003, 004, 1.75, 2, 3.50);

INSERT INTO ORDER_ITEM
          (OrderNumber, ItemID,  Price, ItemQuantity,
           ExtendedPrice)
          VALUES (
          00004, 002, 3.75, 1, 3.75);

INSERT INTO ORDER_ITEM
          (OrderNumber, ItemID, Price, ItemQuantity,
           ExtendedPrice)
          VALUES (
          00004, 003, 2.50, 2, 5.00);

INSERT INTO ORDER_ITEM
          (OrderNumber, ItemID, Price, ItemQuantity,
           ExtendedPrice)
          VALUES (
          00004, 006, 2.00, 1, 2.00);

INSERT INTO ORDER_ITEM
          (OrderNumber, ItemID, Price, ItemQuantity,
           ExtendedPrice)
          VALUES (
          00005, 007, 3.00, 2, 6.00);
```

/*   INSERT INTO INGREDIENT TABLE   */

```
Create Sequence seqIngredientID Increment by 1 start with 101;
INSERT INTO INGREDIENT
        (IngredientID, IngredientName, IngredientCategory, IngredientCost, QuantityOnHand)
        VALUES(seqIngredientID.NextVal, 'Ethical Coffee Beans', 'Coffee Beans', 0.55, 30);

INSERT INTO INGREDIENT
        (IngredientID, IngredientName, IngredientCategory, IngredientCost, QuantityOnHand)
        VALUES(seqIngredientID.NextVal, 'Whole Milk', 'Dairy', 0.25, 15);

INSERT INTO INGREDIENT
        (IngredientID, IngredientName, IngredientCategory, IngredientCost, QuantityOnHand)
        VALUES(seqIngredientID.NextVal, 'Coffee-mate Creamer', 'Coffee Creamers', 0.15, 15);
```

```
INSERT INTO INGREDIENT
        (IngredientID, IngredientName, IngredientCategory, IngredientCost, QuantityOnHand)
        VALUES(seqIngredientID.NextVal, 'Filtered Water', 'Other', 0.20, 30);

INSERT INTO INGREDIENT
        (IngredientID, IngredientName, IngredientCategory, IngredientCost, QuantityOnHand)
        VALUES(seqIngredientID.NextVal, 'Ice', 'Other', 0.10, 30);
```

/*   INSERT INTO INGREDIENT_USE TABLE   */

```
/* If we were to make this more efficient, we would have to bulk insert using another table to
import it into the database */
INSERT INTO INGREDIENT_USE
        (ItemID, IngredientID, AmountIngredientUsed)
        VALUES(001, 101, 3);

INSERT INTO INGREDIENT_USE
        (ItemID, IngredientID, AmountIngredientUsed)
        VALUES(001, 104, 1);

INSERT INTO INGREDIENT_USE
        (ItemID, IngredientID, AmountIngredientUsed)
        VALUES(002, 101, 2);
```

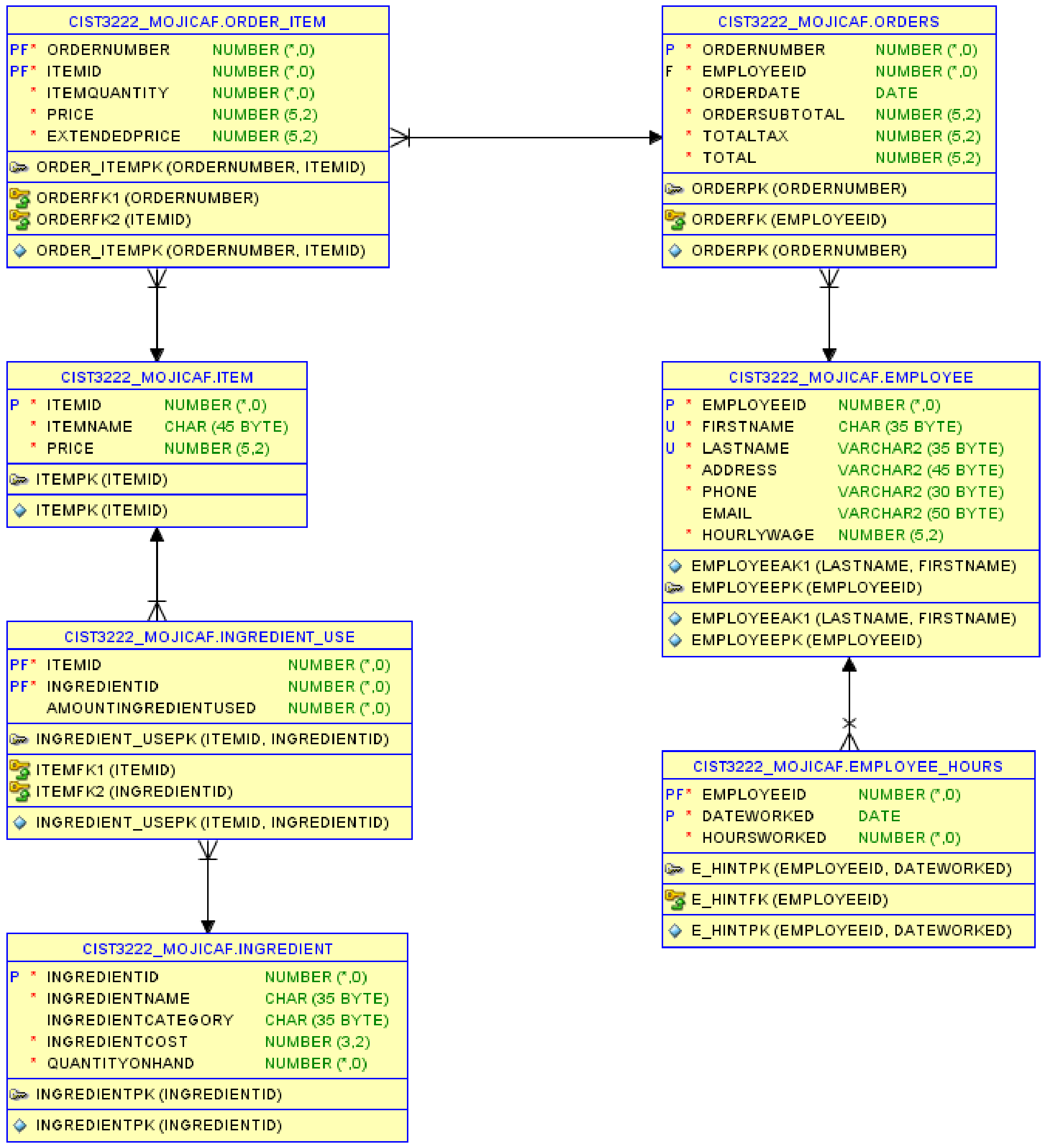
```
INSERT INTO INGREDIENT_USE
        (ItemID, IngredientID, AmountIngredientUsed)
        VALUES(002, 103, 1);

INSERT INTO INGREDIENT_USE
        (ItemID, IngredientID, AmountIngredientUsed)
        VALUES(002, 104, 1);

INSERT INTO INGREDIENT_USE
        (ItemID, IngredientID, AmountIngredientUsed)
        VALUES(002, 105, 3);

INSERT INTO INGREDIENT_USE
        (ItemID, IngredientID, AmountIngredientUsed)
        VALUES(003, 101, 1);

INSERT INTO INGREDIENT_USE
        (ItemID, IngredientID, AmountIngredientUsed)
        VALUES(003, 104, 1);

INSERT INTO INGREDIENT_USE
        (ItemID, IngredientID, AmountIngredientUsed)
```

VALUES(004, 102, 3);

/* COMMIT STATEMENT (FINAL STATEMENT) */

COMMIT;

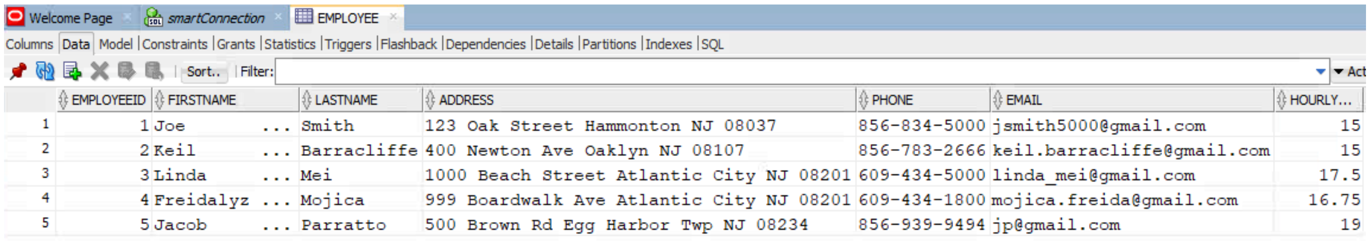
3. Screenshot of the ER diagram in SQL Developer



```
CIST3222_MOJICAF.ORDER_ITEM
PF* ORDERNUMBER        NUMBER (*,0)
PF* ITEMID             NUMBER (*,0)
  * ITEMQUANTITY       NUMBER (*,0)
  * PRICE              NUMBER (5,2)
  * EXTENDEDPRICE      NUMBER (5,2)
ORDER_ITEMPK (ORDERNUMBER, ITEMID)
ORDERFK1 (ORDERNUMBER)
ORDERFK2 (ITEMID)
ORDER_ITEMPK (ORDERNUMBER, ITEMID)

CIST3222_MOJICAF.ORDERS
P * ORDERNUMBER        NUMBER (*,0)
F * EMPLOYEEID         NUMBER (*,0)
  * ORDERDATE          DATE
  * ORDERSUBTOTAL      NUMBER (5,2)
  * TOTALTAX           NUMBER (5,2)
  * TOTAL              NUMBER (5,2)
ORDERPK (ORDERNUMBER)
ORDERFK (EMPLOYEEID)
ORDERPK (ORDERNUMBER)

CIST3222_MOJICAF.ITEM
P * ITEMID             NUMBER (*,0)
  * ITEMNAME           CHAR (45 BYTE)
  * PRICE              NUMBER (5,2)
ITEMPK (ITEMID)
ITEMPK (ITEMID)

CIST3222_MOJICAF.EMPLOYEE
P * EMPLOYEEID         NUMBER (*,0)
U * FIRSTNAME          CHAR (35 BYTE)
U * LASTNAME           VARCHAR2 (35 BYTE)
  * ADDRESS            VARCHAR2 (45 BYTE)
  * PHONE              VARCHAR2 (30 BYTE)
    EMAIL              VARCHAR2 (50 BYTE)
  * HOURLYWAGE         NUMBER (5,2)
EMPLOYEEAK1 (LASTNAME, FIRSTNAME)
EMPLOYEEPK (EMPLOYEEID)
EMPLOYEEAK1 (LASTNAME, FIRSTNAME)
EMPLOYEEPK (EMPLOYEEID)

CIST3222_MOJICAF.INGREDIENT_USE
PF* ITEMID                  NUMBER (*,0)
PF* INGREDIENTID            NUMBER (*,0)
    AMOUNTINGREDIENTUSED    NUMBER (*,0)
INGREDIENT_USEPK (ITEMID, INGREDIENTID)
ITEMFK1 (ITEMID)
ITEMFK2 (INGREDIENTID)
INGREDIENT_USEPK (ITEMID, INGREDIENTID)

CIST3222_MOJICAF.EMPLOYEE_HOURS
PF* EMPLOYEEID         NUMBER (*,0)
P * DATEWORKED         DATE
  * HOURSWORKED        NUMBER (*,0)
E_HINTPK (EMPLOYEEID, DATEWORKED)
E_HINTFK (EMPLOYEEID)
E_HINTPK (EMPLOYEEID, DATEWORKED)

CIST3222_MOJICAF.INGREDIENT
P * INGREDIENTID        NUMBER (*,0)
  * INGREDIENTNAME      CHAR (35 BYTE)
    INGREDIENTCATEGORY  CHAR (35 BYTE)
  * INGREDIENTCOST      NUMBER (3,2)
  * QUANTITYONHAND      NUMBER (*,0)
INGREDIENTPK (INGREDIENTID)
INGREDIENTPK (INGREDIENTID)
```

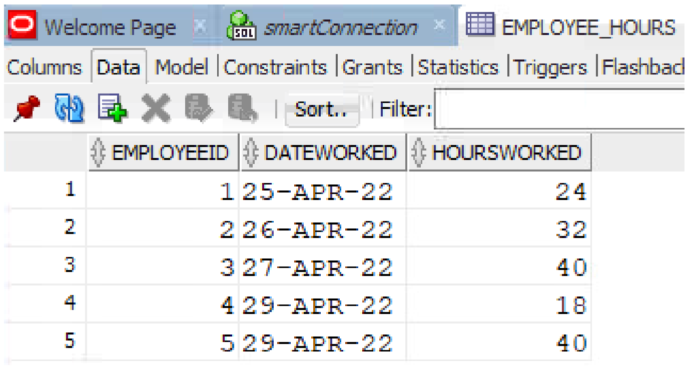
4. Screenshot of the Tables in SQL Developer

EMPLOYEE TABLE


Welcome Page
smartConnection
EMPLOYEE
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:
Act

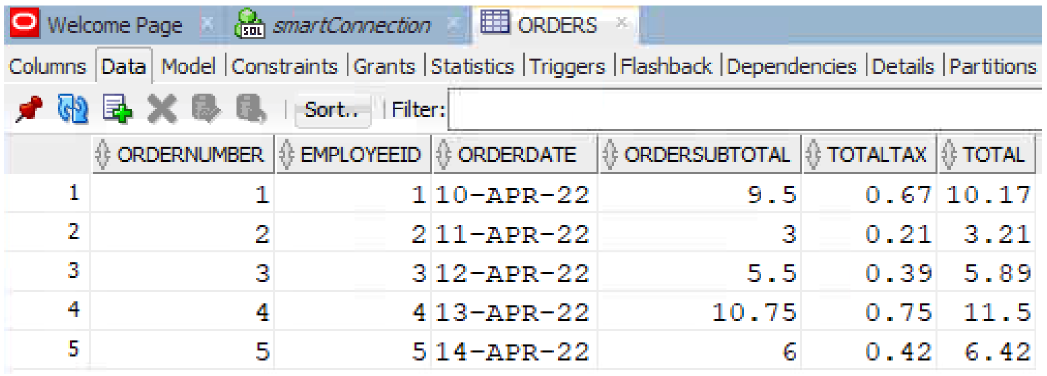
	EMPLOYEEID	FIRSTNAME	LASTNAME	ADDRESS	PHONE	EMAIL	HOURLY...
1	1	Joe ...	Smith	123 Oak Street Hammonton NJ 08037	856-834-5000	jsmith5000@gmail.com	15
2	2	Keil ...	Barracliffe	400 Newton Ave Oaklyn NJ 08107	856-783-2666	keil.barracliffe@gmail.com	15
3	3	Linda ...	Mei	1000 Beach Street Atlantic City NJ 08201	609-434-5000	linda_mei@gmail.com	17.5
4	4	Freidalyz ...	Mojica	999 Boardwalk Ave Atlantic City NJ 08201	609-434-1800	mojica.freida@gmail.com	16.75
5	5	Jacob ...	Parratto	500 Brown Rd Egg Harbor Twp NJ 08234	856-939-9494	jp@gmail.com	19

EMPLOYEE_HOURS TABLE


Welcome Page
smartConnection
EMPLOYEE_HOURS
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashbacl
Sort..
Filter:

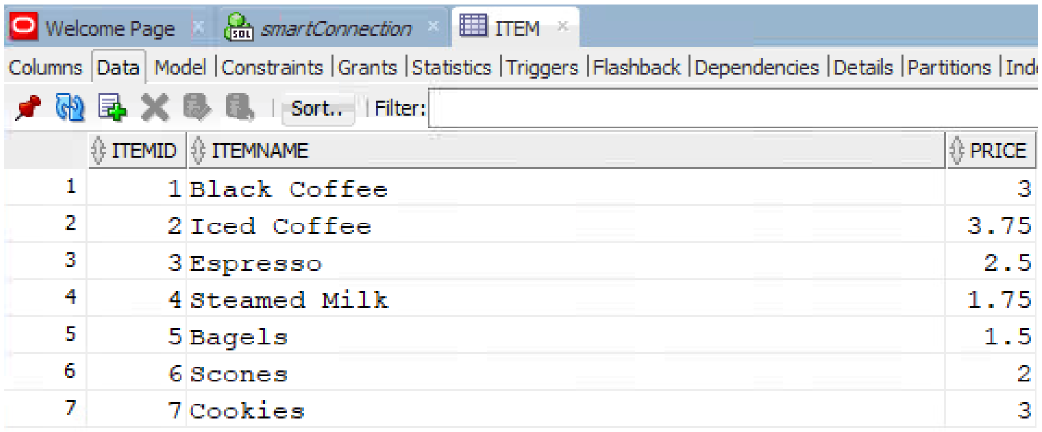
	EMPLOYEEID	DATEWORKED	HOURSWORKED
1	1	25-APR-22	24
2	2	26-APR-22	32
3	3	27-APR-22	40
4	4	29-APR-22	18
5	5	29-APR-22	40

ORDERS TABLE


Welcome Page
smartConnection
ORDERS
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions
Sort..
Filter:

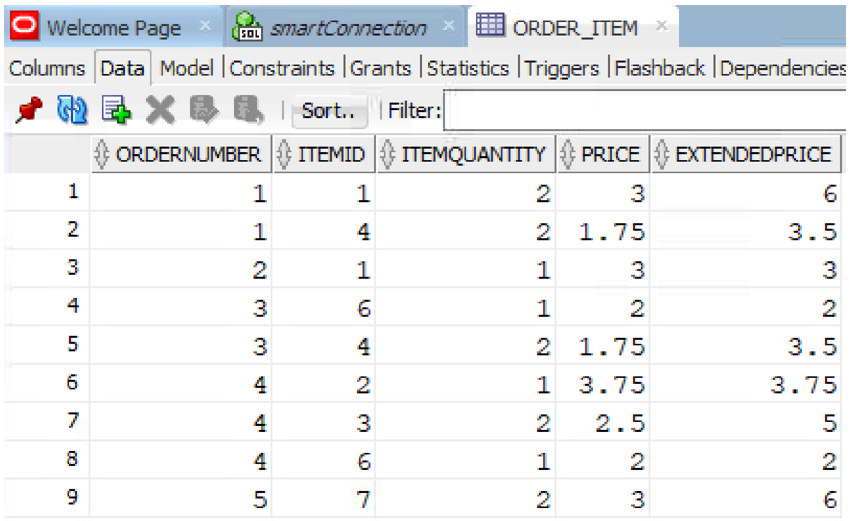
	ORDERNUMBER	EMPLOYEEID	ORDERDATE	ORDERSUBTOTAL	TOTALTAX	TOTAL
1	1	1	10-APR-22	9.5	0.67	10.17
2	2	2	11-APR-22	3	0.21	3.21
3	3	3	12-APR-22	5.5	0.39	5.89
4	4	4	13-APR-22	10.75	0.75	11.5
5	5	5	14-APR-22	6	0.42	6.42

ITEM TABLE



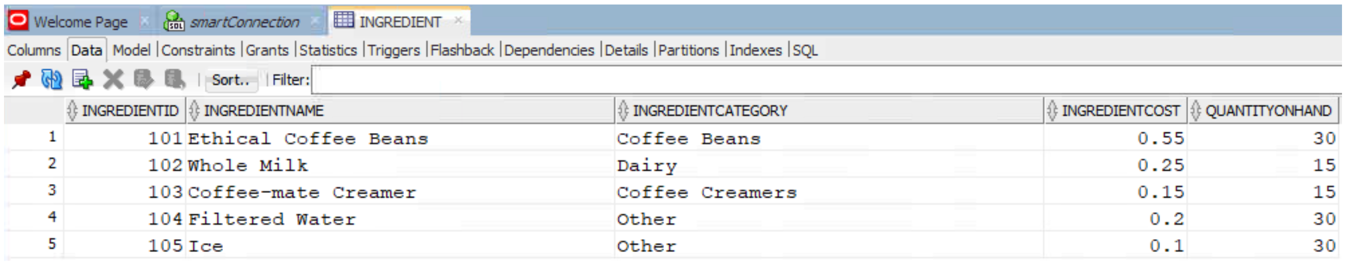
	ITEMID	ITEMNAME	PRICE
1	1	Black Coffee	3
2	2	Iced Coffee	3.75
3	3	Espresso	2.5
4	4	Steamed Milk	1.75
5	5	Bagels	1.5
6	6	Scones	2
7	7	Cookies	3

ORDER_ITEM TABLE



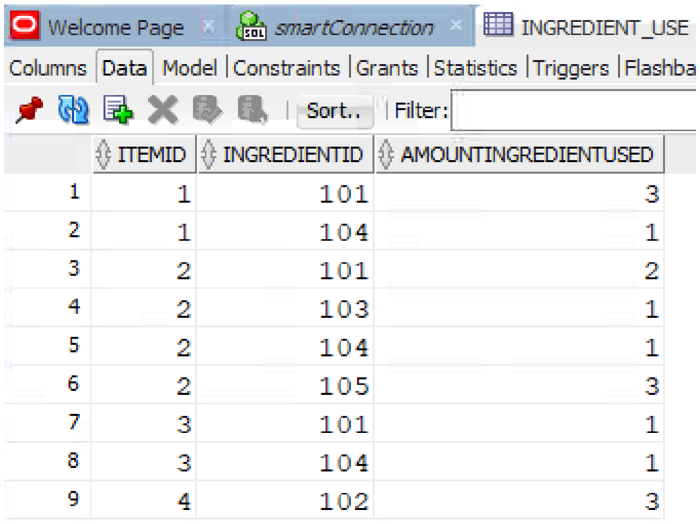
	ORDERNUMBER	ITEMID	ITEMQUANTITY	PRICE	EXTENDEDPRICE
1	1	1	2	3	6
2	1	4	2	1.75	3.5
3	2	1	1	3	3
4	3	6	1	2	2
5	3	4	2	1.75	3.5
6	4	2	1	3.75	3.75
7	4	3	2	2.5	5
8	4	6	1	2	2
9	5	7	2	3	6

INGREDIENT TABLE



	INGREDIENTID	INGREDIENTNAME	INGREDIENTCATEGORY	INGREDIENTCOST	QUANTITYONHAND
1	101	Ethical Coffee Beans	Coffee Beans	0.55	30
2	102	Whole Milk	Dairy	0.25	15
3	103	Coffee-mate Creamer	Coffee Creamers	0.15	15
4	104	Filtered Water	Other	0.2	30
5	105	Ice	Other	0.1	30

INGREDIENT_USE TABLE



	ITEMID	INGREDIENTID	AMOUNTINGREDIENTUSED
1	1	101	3
2	1	104	1
3	2	101	2
4	2	103	1
5	2	104	1
6	2	105	3
7	3	101	1
8	3	104	1
9	4	102	3