Relational schema

Entities:

Employee(eid, fName, lName, shift, salary)
Chef(eid, rank) id foreign key referencing Employee
Manager(eid) id foreign key referencing Employee
Waiter(eid) id foreign key referencing Employee
Host(eid) id f;oreign key referencing Employee
Ingredient(name, stock)
MenuItem(name, description, price)
Supplier(supplierId, name, email, phone)
Table(tableNumber, size)
Bill(billId, date, time, amt_paid)
Reservation(resId, fName, lName, date, time, size, host_id) host_id foreign key referencing Host NOT NULL

Relationships

Seats(tableNumber, host_id, time) table_number foreign key referencing Table
ServesCheque(tableNumber, billId, waiter_id) tableNumber foreign key referencing Table, billId foreign key referencing Bill
Assignment(tableNumber, resId) tableNumber foreign key referencing Table, resId foreign key referencing Reservation
Attends(waiterId, tableNumber) waiterId foreign key referencing Waiter, tableNumber foreign key referencing Table
Contains(itemName, billId) itemName foreign key referencing MenuItem, billId foreign key referencing Bill
Modifies(chefId, managerId, itemName) chefId foreign key referencing Chef, managerId foreign key referencing Manager, itemName foreign key referencing MenuItem
MakesUp(ingredientName, itemName) ingredientName foreign key referencing Ingredient, itemName foreign key referencing MenuItem
Orders(managerId, ingredientName, supplierId, quantity, date) managerId foreign key referencing Manager, ingredientName foreign key referencing Ingredient, supplierId foreign key referencing Supplier

Queries:

--Query 1: Average total sales by month

```
SELECT DISTINCT MONTH, YEAR, SUM(AMTPAID) AS TOTALSALES FROM
(SELECT MONTH(BILL.DATE) AS MONTH, YEAR(BILL.DATE) AS YEAR, AMTPAID
FROM BILL)
GROUP BY MONTH, YEAR
ORDER BY MONTH, YEAR;
```

```
db2 => SELECT DISTINCT MONTH, YEAR, SUM(AMTPAID) AS TOTALSALES FROM (SELECT MONTH(BILL.DATE) AS MONTH, YEAR(BILL.DATE) A
S YEAR, AMTPAID FROM BILL) GROUP BY MONTH, YEAR ORDER BY MONTH, YEAR;

MONTH       YEAR        TOTALSALES
----------- ----------- --------------------------------
          3        2024                           4063.

  1 record(s) selected.

db2 =>
```

--Query 2: Suppliers of ingredients in a given menu item
--Menu item: 'Beef Burger'

```
SELECT SUPPLIERID, INGREDIENTNAME FROM ORDERS
WHERE INGREDIENTNAME IN (SELECT INGREDIENTNAME FROM MAKESUP WHERE
ITEMNAME = 'Beef Burger')
ORDER BY ORDERS.SUPPLIERID;
```

```
db2 => SELECT SUPPLIERID, INGREDIENTNAME FROM ORDERS WHERE INGREDIENTNAME IN (SELECT INGREDIENTNAME FROM MAKESUP WHERE I
TEMNAME = 'Beef Burger') ORDER BY ORDERS.SUPPLIERID;

SUPPLIERID  INGREDIENTNAME
----------- --------------------
          1 Cheese
          1 Onion
          2 Tomato
          2 Lettuce
          3 Tomato
          5 Cheese
          5 Lettuce

  7 record(s) selected.
```

--Query 3: Average wage by employee type (manager, chef, host, or waiter)

```
SELECT AVG(WAGE) AS AVERAGEWAGE, ETYPE
FROM (
SELECT WAGE,
CASE
        WHEN EID IN (SELECT EID FROM MANAGER) THEN 'Manager'
        WHEN EID IN (SELECT EID FROM CHEF) THEN 'Chef'
```

```
        WHEN EID IN (SELECT EID FROM HOST) THEN 'Host'
        WHEN EID IN (SELECT EID FROM WAITER) THEN 'Waiter'
        ELSE 'Unknown'
END AS ETYPE
FROM Employee
)
GROUP BY ETYPE
ORDER BY AVG(WAGE);
```

```
db2 => SELECT AVG(WAGE) AS AVERAGEWAGE, ETYPE FROM (SELECT WAGE, CASE WHEN EID IN (SELECT EID FROM MANAGER) THEN 'Manage
r' WHEN EID IN (SELECT EID FROM CHEF) THEN 'Chef' WHEN EID IN (SELECT EID FROM HOST) THEN 'Host' WHEN EID IN (SELECT EID
 FROM WAITER) THEN 'Waiter' ELSE 'Unknown' END AS ETYPE FROM Employee ) GROUP BY ETYPE ORDER BY AVG(WAGE);

AVERAGEWAGE                       ETYPE
--------------------------------- -------
    14.600000000000000000000000000 Host
    15.000000000000000000000000000 Manager
    15.000000000000000000000000000 Waiter
    15.200000000000000000000000000 Chef
                                   - Unknown

  5 record(s) selected.

db2 =>
```

--Query 4: Menu items modified by a given employee ID (id = 11)

```
SELECT NAME, DESCRIPTION, PRICE FROM MENUITEM
INNER JOIN MODIFIES ON MENUITEM.NAME = MODIFIES.ITEMNAME
WHERE MODIFIES.CHEFID = 11 OR MODIFIES.MANAGERID = 11;
```

```
db2 => SELECT NAME, DESCRIPTION, PRICE FROM MENUITEM INNER JOIN MODIFIES ON MENUITEM.NAME = MODIFIES.ITEMNAME WHERE MODI
FIES.CHEFID = 11 OR MODIFIES.MANAGERID = 11;

NAME                                               DESCRIPTION
                                  PRICE
-------------------------------------------------- ------------------------------------------------------------------
------------------------------ -------
Beef Burger                                        Classic beef burger with cheese and fries. Contains gluten and dairy.
                                     23.
Beef Burger                                        Classic beef burger with cheese and fries. Contains gluten and dairy.
                                     23.

  2 record(s) selected.
```

--Query 5: Time of the day (morning, afternoon, evening) by total sales
--Shows which time of the day that the restaurant has the most sales

```
SELECT
TIMEOFDAY, SUM(AMTPAID) AS TOTALSALES FROM
(SELECT
CASE
        WHEN TIME < ‘12:00:00’ THEN 'Morning'
```

WHEN TIME < ‘16:00:00’’ THEN 'Afternoon'
WHEN TIME >= ‘16:00:00’ THEN 'Evening'
END AS TIMEOFDAY,
AMTPAID
FROM BILL)
GROUP BY TIMEOFDAY
ORDER BY TOTALSALES;

```
db2 => SELECT TIMEOFDAY, SUM(AMTPAID) AS TOTALSALES FROM (SELECT CASE WHEN TIME < '12:00:00' THEN 'Morning' WHEN TIME <
'16:00:00' THEN 'Afternoon' WHEN TIME >= '16:00:00' THEN 'Evening' END AS TIMEOFDAY, AMTPAID FROM BILL) GROUP BY TIMEOFD
AY ORDER BY TOTALSALES;

TIMEOFDAY TOTALSALES
--------- ---------------------------------
Morning                            645.
Afternoon                         1370.
Evening                           2048.

  3 record(s) selected.

db2 =>
```

6. SQL Modifications

Mod1

a) Two of our suppliers (supplierid 1 and 3, respectively) have contacted us and said that they will be changing their phone numbers. Thus this modification keeps the contact information for our suppliers up to date by updating the relevant phone numbers and leaving the other ones as is
b) UPDATE Supplier SET phone = CASE WHEN supplierid = 1 THEN ‘1023456789’ WHEN supplierid = 3 then ‘2100876542’ ELSE phone END;
c)

```
db2 ⇒ select * from supplier

SUPPLIERID  NAME                                               EMAIL                                              PHONE
----------- -------------------------------------------------- -------------------------------------------------- ----------
          1 Fresh Farms                                        fresh@farm.com                                     1234567890
          2 Dairy Best                                         dairy@best.com                                     2345678901
          3 Meat Co.                                           meat@co.com                                        3456789012
          4 Veggie Mart                                        veggie@mart.com                                    4567890123
          5 Bakers Hub                                         bakers@hub.com                                     5678901234

  5 record(s) selected.
```

```
db2 ⇒  update supplier set phone = case when supplierid = 1 then '1023456789' when supplierid = 3 then '2100876543' else phone end;
DB20000I  The SQL command completed successfully.
db2 ⇒ select * from supplier

SUPPLIERID  NAME                                               EMAIL                                              PHONE
----------- -------------------------------------------------- -------------------------------------------------- ----------
          1 Fresh Farms                                        fresh@farm.com                                     1023456789
          2 Dairy Best                                         dairy@best.com                                     2345678901
          3 Meat Co.                                           meat@co.com                                        2100876543
          4 Veggie Mart                                        veggie@mart.com                                    4567890123
          5 Bakers Hub                                         bakers@hub.com                                     5678901234

  5 record(s) selected.
```

Mod2

a) Business has been booming lately so management has decided to increase all employees' wages by 5 dollars. Here, the modification is updating each employee's wage to be 5 dollars more than what it previously was
b) UPDATE Employee SET wage = wage + 5;
c)

```
db2 ⇒ update employee set wage = wage + 5
DB20000I  The SQL command completed successfully.
db2 ⇒ select * from employee

EID         FNAME                LNAME
----------- -------------------- --------------------------
          1 John                 Smith
          2 Emma                 Johnson
          3 Michael              Brown
          4 Sophia               Davis
          5 James                Wilson
          6 Olivia               Martinez
          7 William              Anderson
          8 Ava                  Taylor
          9 Benjamin             Thomas
         10 Isabella             Hernandez
         11 Ethan                Moore
         12 Mia                  Garcia
         13 Alexander            Lopez
         14 Charlotte            Gonzalez
```

```
db2 ⇒ select * from employee;

EID         FNAME              LNAME                      SHIFT      WAGE
----------- ------------------ -------------------------- ---------- -------
          1 John               Smith                      breakfast      15.
          2 Emma               Johnson                    lunch          14.
          3 Michael            Brown                      dinner         16.
          4 Sophia             Davis                      breakfast      15.
          5 James              Wilson                     lunch          14.
          6 Olivia             Martinez                   dinner         16.
          7 William            Anderson                   breakfast      15.
          8 Ava                Taylor                     lunch          14.
          9 Benjamin           Thomas                     dinner         16.
         10 Isabella           Hernandez                  breakfast      15.
         11 Ethan              Moore                      lunch          14.
         12 Mia                Garcia                     dinner         16.
         13 Alexander          Lopez                      breakfast      15.
         14 Charlotte          Gonzalez                   lunch          14.
         15 Daniel             Rodriguez                  dinner         16.
         16 Ella               Perez                      breakfast      15.
         17 Matthew            Harris                     lunch          14.
         18 Scarlett           Clark                      dinner         16.
         19 Henry              Lewis                      breakfast      15.
         20 Amelia             Walker                     lunch          14.

  20 record(s) selected.
```

7. View Creations

View 1

This view serves as a way to quickly access the employees which receive the highest wage ($16).

CREATE VIEW HIGHESTPAIDEMPLOYEES AS
SELECT EID, LNAME, FNAME FROM EMPLOYEE
WHERE WAGE >= 16;

```
db2 => CREATE VIEW HIGHESTPAIDEMPLOYEES AS SELECT EID, LNAME, FNAME FROM EMPLOYEE WHERE WAGE >= 16;
DB20000I  The SQL command completed successfully.
```

Truncated records:

```
db2 => SELECT * FROM HIGHESTPAIDEMPLOYEES LIMIT 5;

EID         LNAME                          FNAME
----------- ------------------------------ --------------------
          3 Brown                          Michael
          6 Martinez                       Olivia
          9 Thomas                         Benjamin
         12 Garcia                         Mia
         15 Rodriguez                      Daniel

  5 record(s) selected.

db2 =>
```

Insert:

```
db2 => INSERT INTO HIGHESTPAIDEMPLOYEES VALUES (21, 'Jacob', 'Stacey')
DB20000I  The SQL command completed successfully.
db2 => SELECT * FROM HIGHESTPAIDEMPLOYEES

EID         LNAME                          FNAME
----------- ------------------------------ --------------------
          3 Brown                          Michael
          6 Martinez                       Olivia
          9 Thomas                         Benjamin
         12 Garcia                         Mia
         15 Rodriguez                      Daniel
         18 Clark                          Scarlett

  6 record(s) selected.
```

The insert statement claims to have executed successfully, but the inserted values do not appear in the view.

```
db2 => SELECT * FROM EMPLOYEE WHERE LNAME = 'Jacob';

EID         FNAME                LNAME                          SHIFT      WAGE
----------- -------------------- ------------------------------ ---------- -------
         21 Stacey               Jacob                          -                -

  1 record(s) selected.

db2 =>
```

The inserted values do however appear in the source table for the view, missing the column values that were not part of the view.

This is possible because this table is an insertable view. This means that values can be inserted using the view definition. This is the case because at least one column of the view is updateable and since the fullselect of the view does not include a UNION ALL.
https://www.ibm.com/docs/en/db2/11.5?topic=views-insertable

View 2

This view provides the supplier ID along with the ingredient name and the quantity of the ingredient in stock. This can be useful when it comes to determining which ingredients need to be re-ordered in the near future.

```
CREATE VIEW SUPPLIERIDINGREDIENTSTOCK AS
SELECT SUPPLIERID, INGREDIENTNAME, QUANTITY
FROM ORDERS INNER JOIN INGREDIENT
ON ORDERS.INGREDIENTNAME = INGREDIENT.NAME
ORDER BY INGREDIENTNAME;
```

```
db2 => CREATE VIEW SUPPLIERIDINGREDIENTSTOCK AS SELECT SUPPLIERID, INGREDIENTNAME, QUANTITY FROM ORDERS INNER JOIN INGRE
DIENT ON ORDERS.INGREDIENTNAME = INGREDIENT.NAME;
DB20000I  The SQL command completed successfully.
```

Truncated records:

```
db2 => SELECT * FROM SUPPLIERIDINGREDIENTSTOCK LIMIT 5;

SUPPLIERID  INGREDIENTNAME       QUANTITY
----------- -------------------- --------
          1 Chicken                50.
          2 Lettuce               100.
          3 Tomato                 75.
          4 Pasta                  60.
          5 Cheese                 40.

  5 record(s) selected.

db2 =>
```

Insert:

```
db2 => INSERT INTO SUPPLIERIDINGREDIENTSTOCK VALUES (1, 'Carrot', 0)
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0150N  The target fullselect, view, typed table, materialized query table,
range-clustered table, or staging table in the INSERT, DELETE, UPDATE, MERGE,
or TRUNCATE statement is a target for which the requested operation is not
permitted.  SQLSTATE=42807
db2 =>
```

In this case the insert failed.

This view is a read-only view as it was constructed using more than one table.
https://www.ibm.com/docs/en/db2/11.5?topic=views-read-only

8. Check constraints

Check1

a) This constraint enforces the rule that no employee can have a wage that is below 13 dollars (the minimum wage by law in our restaurant's world)

b)

```sql
CREATE TABLE Employee
(
eid INTEGER PRIMARY KEY NOT NULL,
fName VARCHAR(20),
lName VARCHAR(30),
shift VARCHAR(10),
wage DECIMAL CHECK(wage > 13.0)
);
```

c)

```
db2 ⇒ insert into employee (eid, fname, lname, shift, wage) values (21, 'Jack', 'Daniels', 'breakfast', 10.0);
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0545N  The requested operation is not allowed because a row does not
satisfy the check constraint "CS4216109.EMPLOYEE.SQL250307154049520".
```

Check2

a) This constraint makes sure that no ingredient is listed as having “negative” stock in our database

b)

```sql
CREATE TABLE Ingredient
(
name VARCHAR(20) PRIMARY KEY NOT NULL,
stock DECIMAL CHECK(stock ⩾ 0.0)
);
```

c)

```
db2 ⇒ insert into ingredient (name, stock) values ('Cream', -1);
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0545N  The requested operation is not allowed because a row does not
satisfy the check constraint "CS4216109.INGREDIENT.SQL250307154049690".
SQLSTATE=23513
```

10. Team contributions

There were 2 meetings arranged to complete the P2 deliverable. The breakdown of team contributions is as follows:

- Aryan: Table creation, data insertion, data modification, constraint implementation
- Gabrielle: Writing and execution of SQL queries, creation, usage, and updating of views
- Alan: Table creation, data insertion, data modification, constraint implementation