Online Food Ordering System

Team C

Aaron Merrell, Anusha Foo-Singh, Gyananand Singh, and Vasavi Hegde

MSIT 630 Fall 2022 Database Systems

Dr. Mary Harward

October 16th, 2022

Table of Contents

Online Food Ordering System (OFOS)

Overview

The Covid-19 pandemic over the last few years has directly influenced the demand for online food ordering. Takeaways online or pay-for-delivery are solutions many restaurants employ as viable solutions.

The online food ordering system proposed by our team will consist of two main components. First, an application allows customers to view the online menu and quickly place an order per their preference. Second, an administrative interface that enables all orders to be received and fulfilled by the restaurant.

The payments will be accepted online by credit card payment or PayPal. All details of users will be securely maintained to facilitate payment details and customer privacy.

The online food ordering system will function as follows;

- Customers will connect to the food ordering application, and they can navigate to the menu and select orders of their choice.
- The orders selected will be added to the shopping cart. The customers can view the cart and update quantities and items as desired.
- A payment method will be available to customers, such as paying via PayPay or credit card. If via the latter, the customer must sign in or sign up and provide their credit card information.
- Once payment is confirmed, the customer can view the progress of their orders by selecting the view "status" tab.

- The admin user can monitor orders and update their status as required. The admin user manages user account information.

<u>Design and Implementation</u>

E-R Diagram

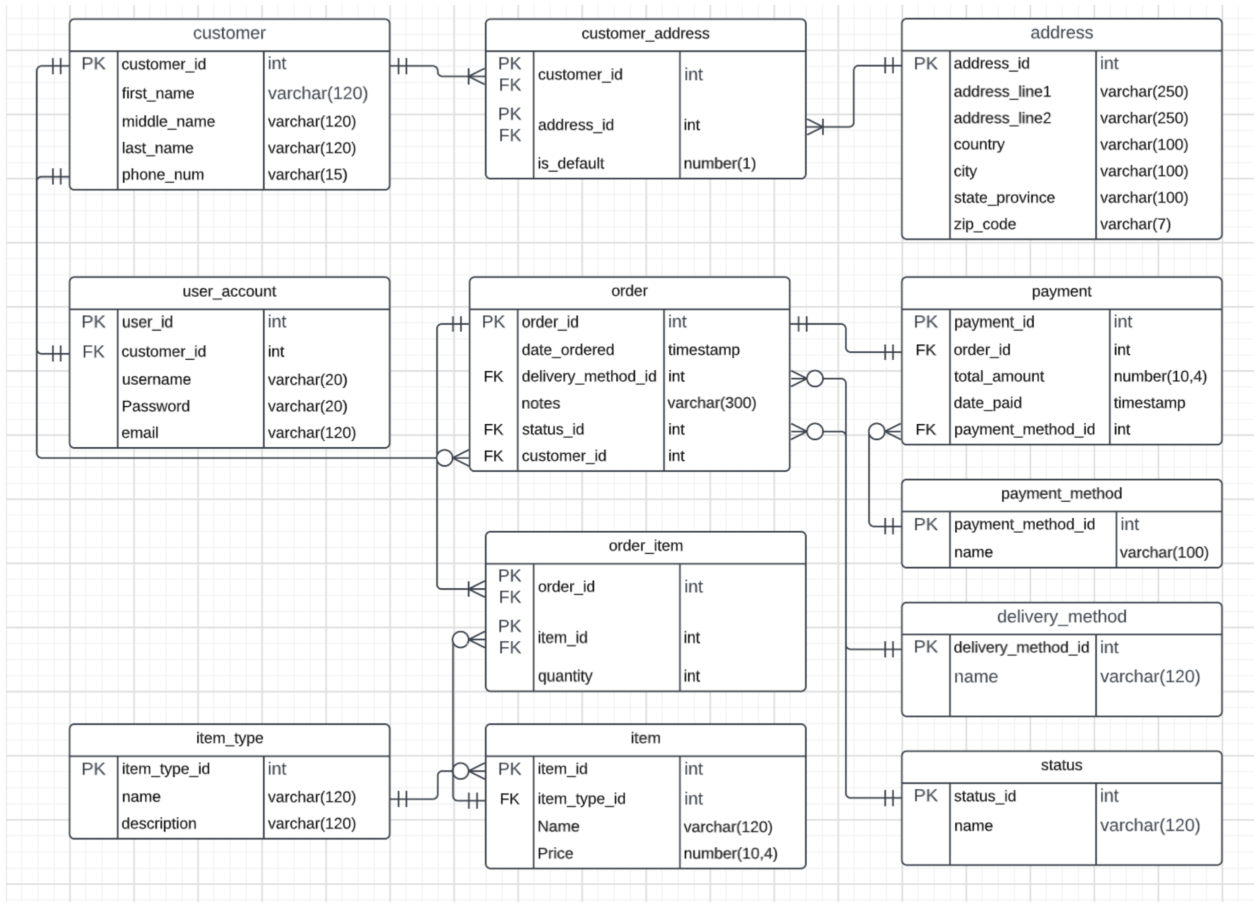


Figure 1: Diagram showing the ER Database Design for OFOS

Relational Database Schema

Entity: delivery_method

The entity “delivery_method” represents the various ways food can be delivered, such as in-store pickup, drop-off location, etc.

Normalization:

BCNF - The entity item_type is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. It is in 3NF because there are no transitive dependencies such that A -> B where A and B are non-prime. Finally, it is in BCNF since A -> B, and A is a super key for all functional dependencies.

Schema:

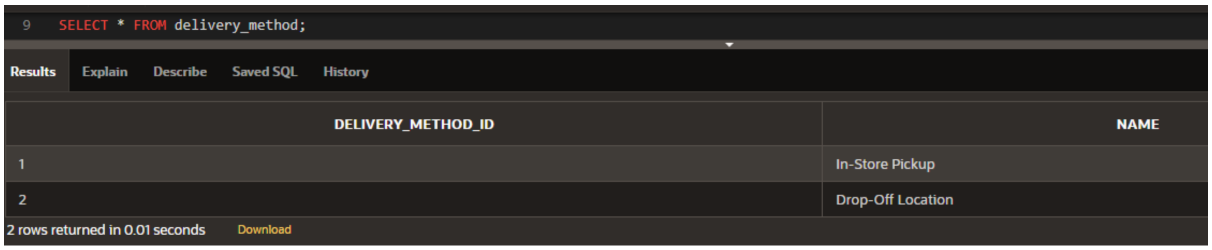
```
delivery_method
(
delivery_method_id int NOT NULL,    -- uniquely identifies each delivery method
name varchar(120) NOT NULL,        -- the name of the delivery method which cannot be more
than 120 characters in length and required
CONSTRAINT pk_delivery_method_id PRIMARY KEY CLUSTERED (delivery_method_id),
CONSTRAINT delivery_method_uk UNIQUE(name),    -- each delivery method name should
be unique
)
```

DDL:

```
create table delivery_method
	(delivery_method_id  int not null,
	 name  varchar(120) not null,
	 primary key (delivery_method_id)
	);
```

Results:

SELECT * FROM delivery_method;



```
SELECT * FROM delivery_method;
```

DELIVERY_METHOD_ID	NAME
1	In-Store Pickup
2	Drop-Off Location

2 rows returned in 0.01 seconds

Entity: item_type

The entity “item_type” represents the various types of the food items, such as drinks, desserts, pasta, etc.

Normalization:

BCNF - The entity item_type is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. It is in 3NF because there are no transitive dependencies such that A -> B where A and B are non-prime. Finally, it is in BCNF since A -> B, and A is a super key for all functional dependencies.

Schema:

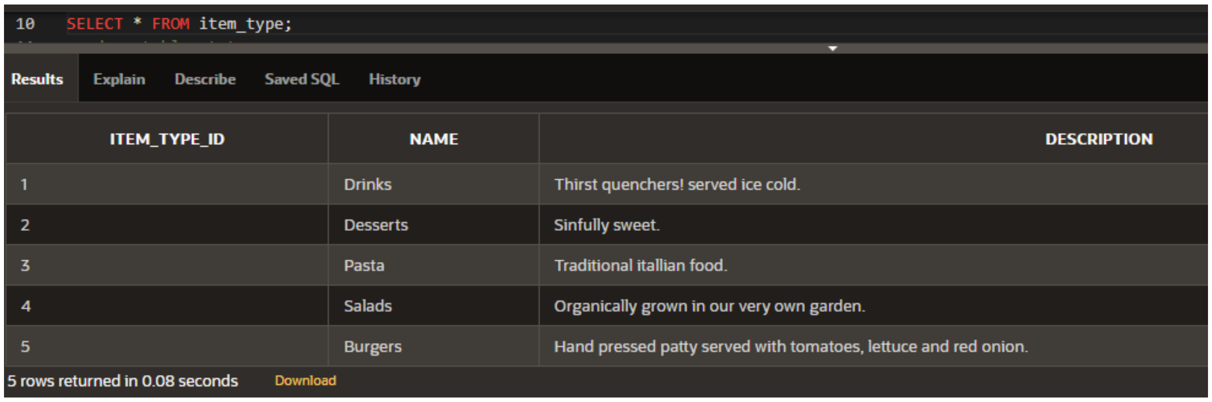
```
item_type

(
item_type_id int NOT NULL,          -- uniquely identifies each item type
name varchar(120) NOT NULL,         -- the name of the item type which cannot be more than 120 characters in length and required
description varchar(300) NOT NULL,    -- a short description of the type which should not be more than 300 characters in length and required
CONSTRAINT pk_item_type_id PRIMARY KEY CLUSTERED (item_type_id),
CONSTRAINT item_type_uk UNIQUE(name),       -- each type name should be unique
)
```

DDL:

```
create table item_type
        (item_type_id  int not null,
         name   varchar(120) not null,
         description    varchar(120) not null,
         primary key (item_type_id)
        );
```

Results:

SELECT * FROM item_type;



```
SELECT * FROM item_type;
```

ITEM_TYPE_ID	NAME	DESCRIPTION
1	Drinks	Thirst quenchers! served ice cold.
2	Desserts	Sinfully sweet.
3	Pasta	Traditional itallian food.
4	Salads	Organically grown in our very own garden.
5	Burgers	Hand pressed patty served with tomatoes, lettuce and red onion.

5 rows returned in 0.08 seconds   Download

Entity: status

The entity “status” represents an order's various statuses, such as order received, preparing, etc.

Normalization:

BCNF - The entity status is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. It is in 3NF because there are no transitive dependencies such that A -> B where A and B are non-prime. Finally, it is in BCNF since A -> B, and A is a super key for all functional dependencies.

Schema:

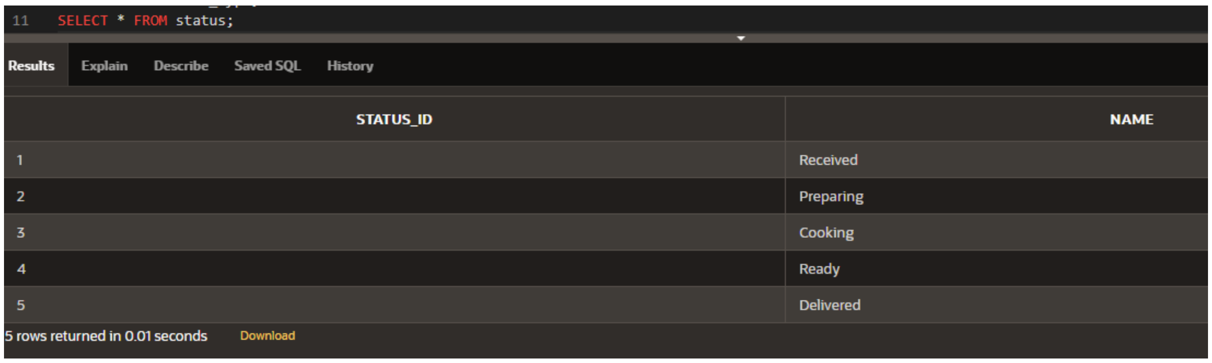
```
status
(
status_id int NOT NULL,         -- uniquely identifies each status
name varchar(120) NOT NULL,         -- the name of the status which cannot be more than 120
characters in length and required
CONSTRAINT pk_status_id PRIMARY KEY CLUSTERED (status_id),
CONSTRAINT status_uk UNIQUE(name),     -- each status should be unique
)
```

DDL:

```
create table status
            (status_id int not null,
             name   varchar(120) not null,
             primary key (status_id)
            );
```

Results:

SELECT * FROM status;



```
SELECT * FROM status;
```

STATUS_ID	NAME
1	Received
2	Preparing
3	Cooking
4	Ready
5	Delivered

5 rows returned in 0.01 seconds

Entity: payment_method

The entity "payment_method" represents how a customer can pay for an order, such as a credit card, PayPal, etc.

Normalization:

BCNF - The entity payment_method is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. It is in 3NF because there are no transitive dependencies such that A -> B where A and B are non-prime. Finally, it is in BCNF since A -> B, and A is a super key for all functional dependencies.

Schema:

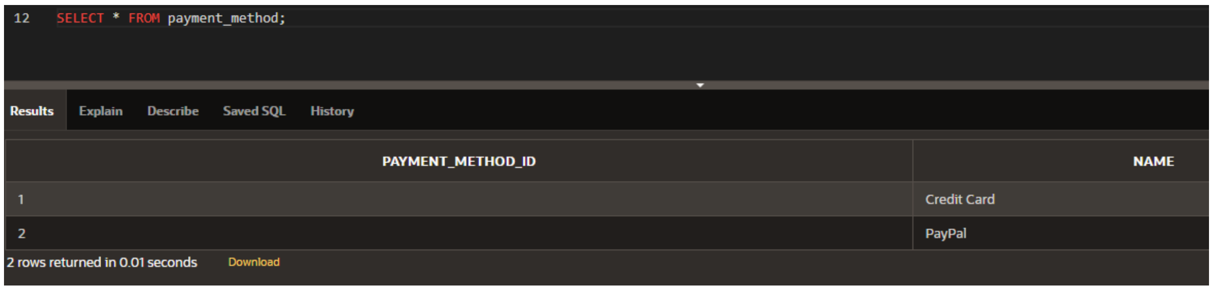
```
 payment_method
(
payment_method_id int NOT NULL,            -- uniquely identifies each payment method
name varchar(120) NOT NULL,         -- the name of the payment method which cannot be more
than 120 characters in length and required
CONSTRAINT pk_payment_method_id PRIMARY KEY CLUSTERED (payment_method_id),
CONSTRAINT payment_method_uk UNIQUE(name),   -- each payment method name should
be unique
)
```

DDL:

```
create table payment_method
          (payment_method_id  int not null,
           name   varchar(100) not null,
           primary key (payment_method_id)
          );
```

Results:

SELECT * FROM payment_method;



```
SELECT * FROM payment_method;
```

PAYMENT_METHOD_ID	NAME
1	Credit Card
2	PayPal

2 rows returned in 0.01 seconds

Entity: order

The entity "order" represents the orders a customer can make for food online.

Normalization:

BCNF - The entity order is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. It is in 3NF because there are no transitive dependencies such that A -> B where A and B are non-prime. Finally, it is in BCNF since for all functional dependencies, A -> B, and A is a super key for all functional dependencies.

Schema:

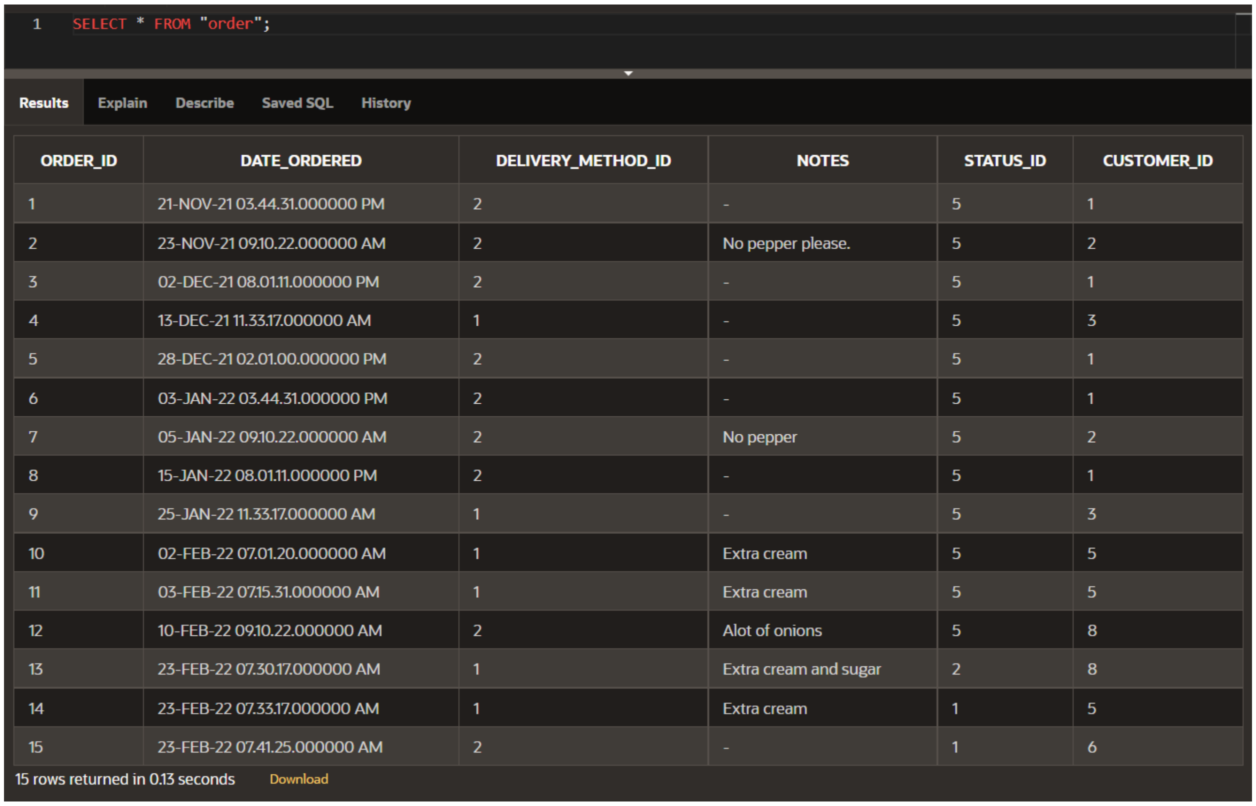
```
order
(
order_id int NOT NULL,        -- uniquely identifies each order
date_ordered timestamp NOT NULL,          -- the required date and time the order was made
delivery_method_id int NOT NULL,          -- the required method of delivery chosen for the
order
notes varchar(300) NULL,      -- any optional notes the customer may have on the order for the
chef which cannot be longer than 300 characters in length
status_id int NOT NULL,                   -- the current status of the order which is required
customer_id int NOT NULL,                 -- the order belongs to a customer, which is required
CONSTRAINT pk_order_id PRIMARY KEY CLUSTERED (order_id),
CONSTRAINT fk_order_delivery_method_id FOREIGN KEY (delivery_method_id)
REFERENCES [delivery_method](delivery_method_id),
CONSTRAINT fk_order_status_id FOREIGN KEY (status_id) REFERENCES
[status](status_id),
CONSTRAINT fk_order_customer_id FOREIGN KEY (customer_id) REFERENCES
[customer](customer_id)
)
```

DDL:

```
create table "order"
        (order_id int not null,
         date_ordered  timestamp not null,
         delivery_method_id   int not null,
         notes   varchar(300) null,
         status_id        int not null,
         customer_id               int not null,
         primary key (order_id),
         foreign key (delivery_method_id) references delivery_method (delivery_method_id)
                on delete set null,
         foreign key (status_id) references status (status_id)
                on delete set null,
         foreign key (customer_id) references customer (customer_id)
                on delete set null
        );
```

Results:

SELECT * FROM "order";



```
SELECT * FROM "order";
```

Results Explain Describe Saved SQL History

ORDER_ID	DATE_ORDERED	DELIVERY_METHOD_ID	NOTES	STATUS_ID	CUSTOMER_ID
1	21-NOV-21 03.44.31.000000 PM	2	-	5	1
2	23-NOV-21 09.10.22.000000 AM	2	No pepper please.	5	2
3	02-DEC-21 08.01.11.000000 PM	2	-	5	1
4	13-DEC-21 11.33.17.000000 AM	1	-	5	3
5	28-DEC-21 02.01.00.000000 PM	2	-	5	1
6	03-JAN-22 03.44.31.000000 PM	2	-	5	1
7	05-JAN-22 09.10.22.000000 AM	2	No pepper	5	2
8	15-JAN-22 08.01.11.000000 PM	2	-	5	1
9	25-JAN-22 11.33.17.000000 AM	1	-	5	3
10	02-FEB-22 07.01.20.000000 AM	1	Extra cream	5	5
11	03-FEB-22 07.15.31.000000 AM	1	Extra cream	5	5
12	10-FEB-22 09.10.22.000000 AM	2	Alot of onions	5	8
13	23-FEB-22 07.30.17.000000 AM	1	Extra cream and sugar	2	8
14	23-FEB-22 07.33.17.000000 AM	1	Extra cream	1	5
15	23-FEB-22 07.41.25.000000 AM	2	-	1	6

15 rows returned in 0.13 seconds Download

Entity: item

The entity "item" represents individual food items the customer can order.

Normalization:

3NF - The entity item is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. Finally, it is in 3NF because there are no transitive dependencies such that A -> B where A and B are non-prime.

Schema:

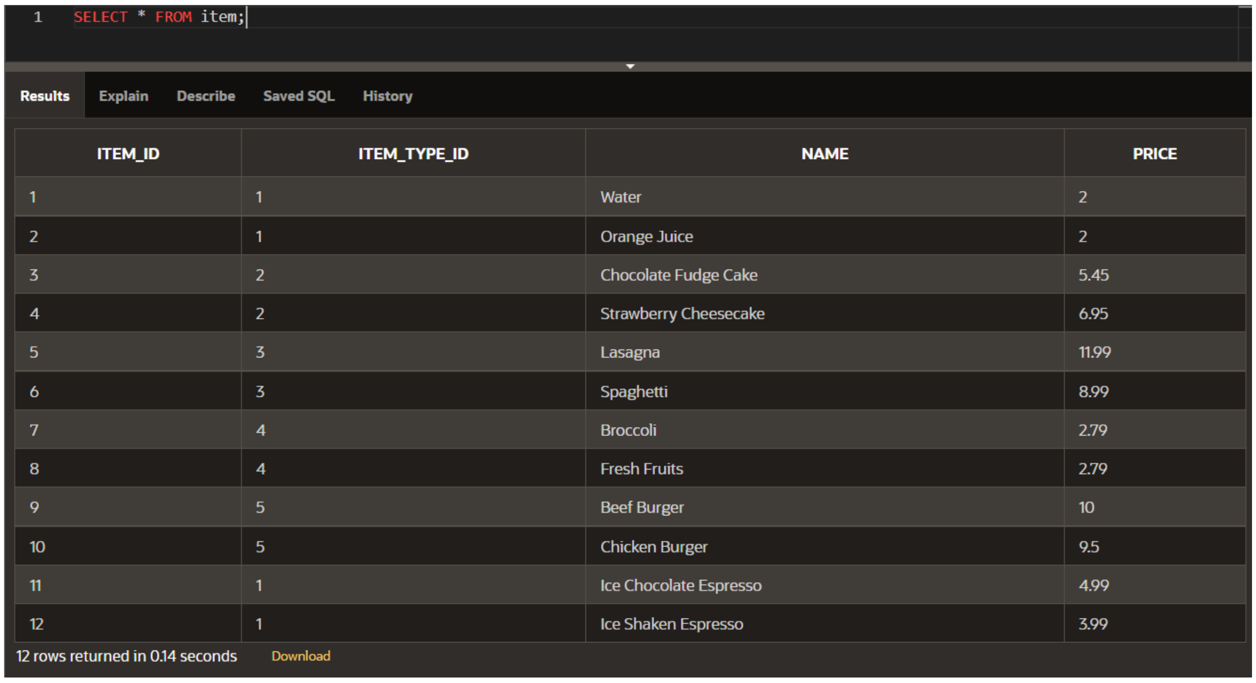
```
item
(
item_id int NOT NULL,    -- uniquely identifies each food item
item_type_id int NOT NULL,          -- each item belongs to a type and is required
name varchar(120) NOT NULL,        -- the name of the item which cannot be longer than 120
characters in length
price number(10,4) NOT NULL,       -- each item must have a cost
CONSTRAINT pk_item_id PRIMARY KEY CLUSTERED (item_id),
CONSTRAINT item_uk UNIQUE(name),              -- each item name should be unique
CONSTRAINT fk_item_item_type_id FOREIGN KEY (item_type_id) REFERENCES
[item_type](item_type_id)
)
```

DDL:

```
create table item
        (item_id        int not null,
    item_type_id        int not null,
        name            varchar(120) not null,
        price           number(10,4) not null,
        primary key (item_id),
        foreign key (item_type_id) references item_type (item_type_id)
                on delete cascade
        );
```

Results:

SELECT * FROM item;



```
SELECT * FROM item;
```

ITEM_ID	ITEM_TYPE_ID	NAME	PRICE
1	1	Water	2
2	1	Orange Juice	2
3	2	Chocolate Fudge Cake	5.45
4	2	Strawberry Cheesecake	6.95
5	3	Lasagna	11.99
6	3	Spaghetti	8.99
7	4	Broccoli	2.79
8	4	Fresh Fruits	2.79
9	5	Beef Burger	10
10	5	Chicken Burger	9.5
11	1	Ice Chocolate Espresso	4.99
12	1	Ice Shaken Espresso	3.99

12 rows returned in 0.14 seconds

Entity: order_item

The entity (bridge) “order_item” allows an order to have many items in it and an item to be a part of many orders.

Normalization:

BCNF - The entity order_item is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. It is in 3NF because there are no transitive dependencies such that A -> B where A and B are non-prime. Finally, it is in BCNF since for all functional dependencies, A -> B, and A is a super key for all functional dependencies.

Schema:

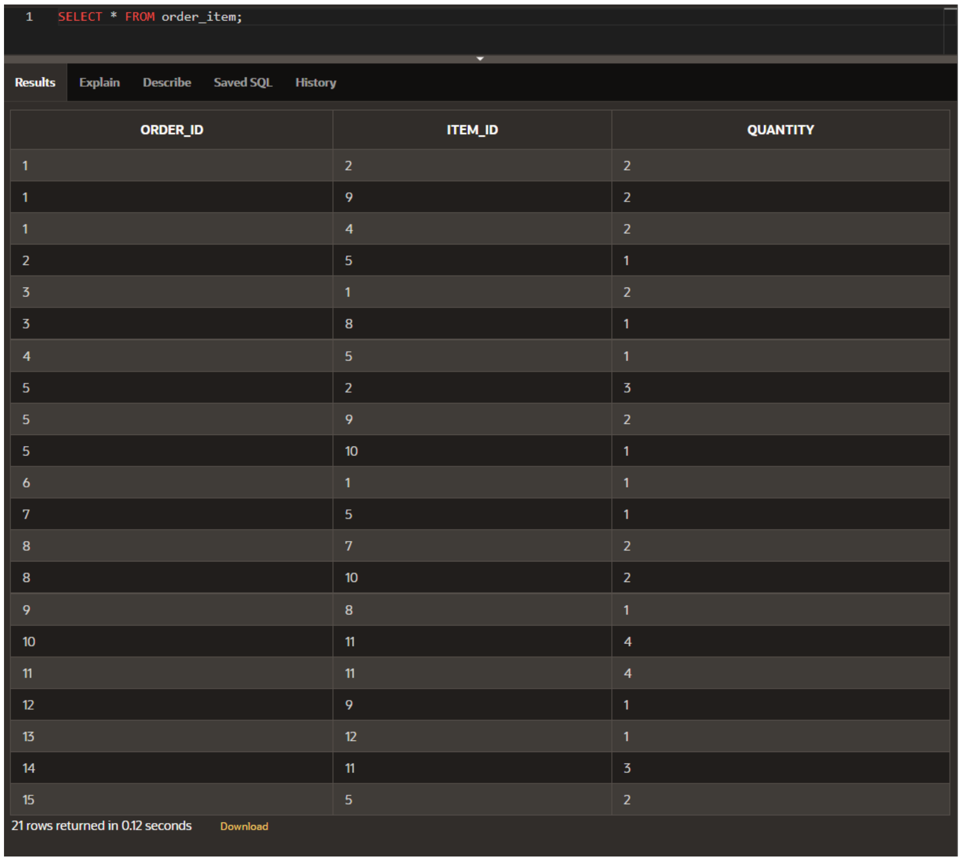
```
order_item
(
order_id int NOT NULL, -- uniquely identifies which order and is part of the primary key
item_id int NOT NULL,    -- uniquely identifies which item and is part of the primary key
quantity int NOT NULL,    -- the amount of the items needed which is required
CONSTRAINT pk_order_item_id PRIMARY KEY CLUSTERED (order_id, item_id),
CONSTRAINT fk_order_item_order_id FOREIGN KEY (order_id) REFERENCES
[order](order_id),
CONSTRAINT fk_order_item_item_id FOREIGN KEY (item_id) REFERENCES
[item](item_id)
)
```

DDL:

```
create table order_item
        (order_id       int not null,
         item_id        int not null,
         quantity       int not null,
         foreign key (order_id) references "order" (order_id)
                on delete cascade,
         foreign key (item_id) references item (item_id)
                on delete cascade
        );
```

Results:

```
SELECT * FROM order_item;
```



```
SELECT * FROM order_item;
```

Results Explain Describe Saved SQL History

ORDER_ID	ITEM_ID	QUANTITY
1	2	2
1	9	2
1	4	2
2	5	1
3	1	2
3	8	1
4	5	1
5	2	3
5	9	2
5	10	1
6	1	1
7	5	1
8	7	2
8	10	2
9	8	1
10	11	4
11	11	4
12	9	1
13	12	1
14	11	3
15	5	2

21 rows returned in 0.12 seconds Download

Entity: payment

The entity “payment” represents the payment details for an order.

Normalization:

3NF - The entity payment is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. Finally, it is in 3NF because there are no transitive dependencies such that A -> B where A and B are non-prime.

Schema:

payment
(

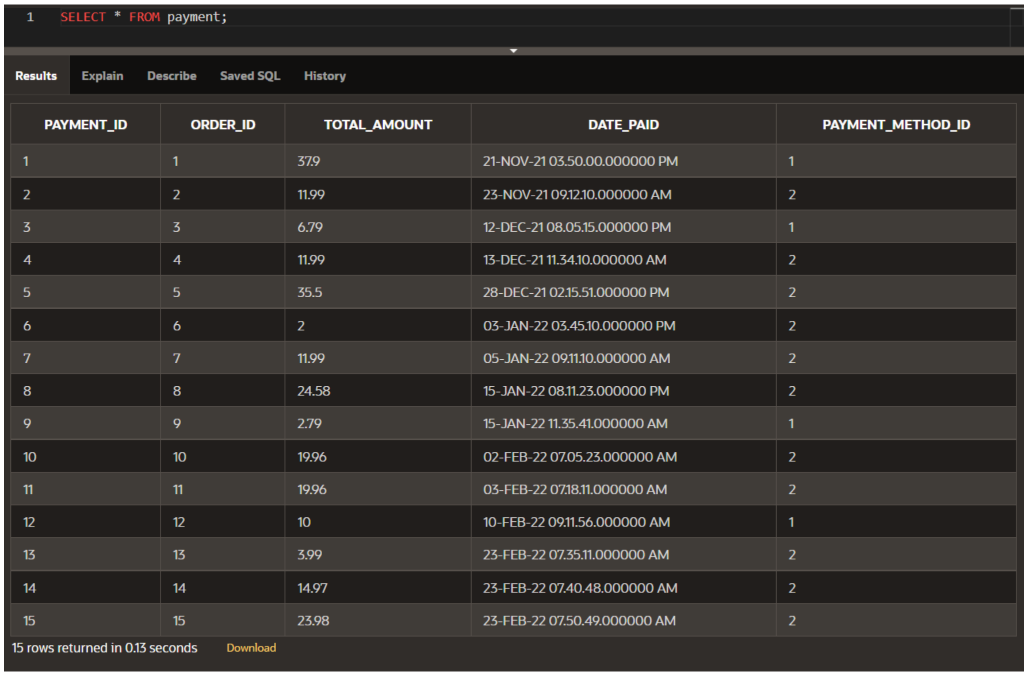
```
payment_id int NOT NULL,            -- uniquely identifies a payment

order_id int NOT NULL,         -- the payment must belong to an order

total_amount number(10,4) NOT NULL,   -- the total paid for the order with precision

date_paid timestamp NOT NULL,     -- the date and time the payment was made, which is
required

payment_method_id int NOT NULL,          -- the required type of payment method used for this
payment

CONSTRAINT pk_payment_id PRIMARY KEY CLUSTERED (payment_id),

CONSTRAINT fk_payment_order_id FOREIGN KEY (order_id) REFERENCES
[order](order_id),

CONSTRAINT fk_payment_payment_method_id FOREIGN KEY (payment_method_id)
REFERENCES [payment_method](payment_method_id)

)
```

DDL:

```
create table payment
          (payment_id    int not null,
           order_id         int not null,
           total_amount  number(10,4) not null,
           date_paid        timestamp not null,
           payment_method_id  int not null,
           primary key (payment_id),
           foreign key (order_id) references "order"  (order_id)
                    on delete set null,
           foreign key (payment_method_id) references payment_method (payment_method_id)
                    on delete set null
           );
```

Results:

SELECT * FROM payment;



```
SELECT * FROM payment;
```

Results | Explain | Describe | Saved SQL | History

PAYMENT_ID	ORDER_ID	TOTAL_AMOUNT	DATE_PAID	PAYMENT_METHOD_ID
1	1	37.9	21-NOV-21 03.50.00.000000 PM	1
2	2	11.99	23-NOV-21 09.12.10.000000 AM	2
3	3	6.79	12-DEC-21 08.05.15.000000 PM	1
4	4	11.99	13-DEC-21 11.34.10.000000 AM	2
5	5	35.5	28-DEC-21 02.15.51.000000 PM	2
6	6	2	03-JAN-22 03.45.10.000000 PM	2
7	7	11.99	05-JAN-22 09.11.10.000000 AM	2
8	8	24.58	15-JAN-22 08.11.23.000000 PM	2
9	9	2.79	15-JAN-22 11.35.41.000000 AM	1
10	10	19.96	02-FEB-22 07.05.23.000000 AM	2
11	11	19.96	03-FEB-22 07.18.11.000000 AM	2
12	12	10	10-FEB-22 09.11.56.000000 AM	1
13	13	3.99	23-FEB-22 07.35.11.000000 AM	2
14	14	14.97	23-FEB-22 07.40.48.000000 AM	2
15	15	23.98	23-FEB-22 07.50.49.000000 AM	2

15 rows returned in 0.13 seconds   Download

Entity: customer

The entity “customer” represents the details of a customer.

Normalization:

BCNF - The entity customer is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. It is in 3NF because there are no transitive dependencies such that A -> B where A and B are non-prime. Finally, it is in BCNF since A -> B, and A is a super key for all functional dependencies.

Schema:

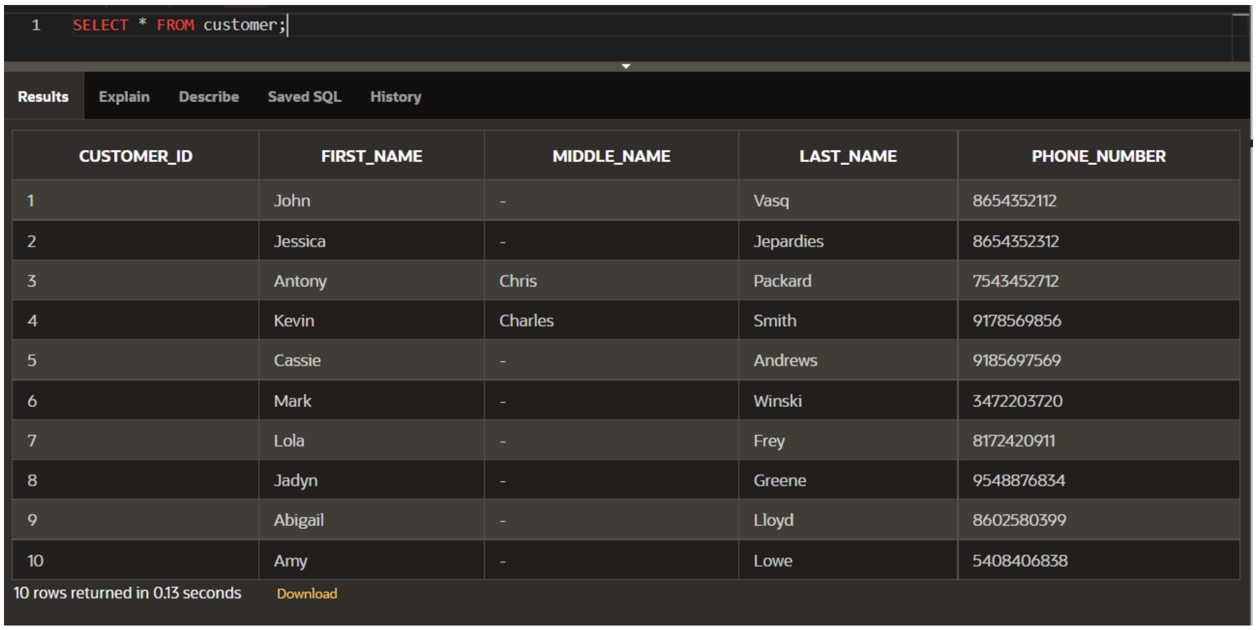
```
customer
(
customer_id int NOT NULL,            -- uniquely identifies each customer
first_name varchar(120) NOT NULL,    -- the first name of the customer, which cannot be longer than 120 characters in length and is required
middle_name varchar(120) NULL,              -- the middle name of the customer, if any
last_name varchar(120) NOT NULL,            -- the last name of the customer, which cannot be longer than 120 characters in length and is required
phone_num varchar(50) NOT NULL,    -- the customer phone number for contact, which is required and has a variable length of up to 50 characters
CONSTRAINT pk_customer_id PRIMARY KEY CLUSTERED (customer_id)
)
```

DDL:

```
create table customer
        (customer_id          int not null,
         first_name           varchar(120) not null,
         middle_name varchar(120) null,
         last_name            varchar(120) not null,
         phone_number         varchar(15) not null,
         primary key (customer_id)
        );
```

Results:

SELECT * FROM customer;



```
SELECT * FROM customer;
```

Results Explain Describe Saved SQL History

CUSTOMER_ID	FIRST_NAME	MIDDLE_NAME	LAST_NAME	PHONE_NUMBER
1	John	-	Vasq	8654352112
2	Jessica	-	Jepardies	8654352312
3	Antony	Chris	Packard	7543452712
4	Kevin	Charles	Smith	9178569856
5	Cassie	-	Andrews	9185697569
6	Mark	-	Winski	3472203720
7	Lola	-	Frey	8172420911
8	Jadyn	-	Greene	9548876834
9	Abigail	-	Lloyd	8602580399
10	Amy	-	Lowe	5408406838

10 rows returned in 0.13 seconds Download

Entity: address

The entity "address" represents the address details for a customer.

Normalization:

2NF - The entity address is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. It violates 3NF since the dependency zip_code -> city, state_province where zip_code is non-prime, making it a transitive dependency.

Schema:

address

(

address_id int NOT NULL, *-- uniquely identifies each address*

address_line1 varchar(250) NOT NULL, *-- the first line of the address, such as street, which is required and has a variable length of up to 250 characters*

address_line2 varchar(250) NULL, *-- the second line of the address, if any, and has a variable length of up to 250 characters*

country varchar(100) NOT NULL, *-- the country the customer lives in, which is required and has a variable length of up to 100 characters*

city varchar(100) NOT NULL, *-- the city the customer lives in, which is required and has a variable length of up to 100 characters*

state_province varchar(100) NOT NULL, *-- the state the customer lives in, which is required and has a variable length of up to 100 characters*

zip_code varchar(10) NOT NULL, *-- the zip code for the address which is required and has a variable length of up to 10 characters*

CONSTRAINT pk_address_id PRIMARY KEY CLUSTERED (address_id)

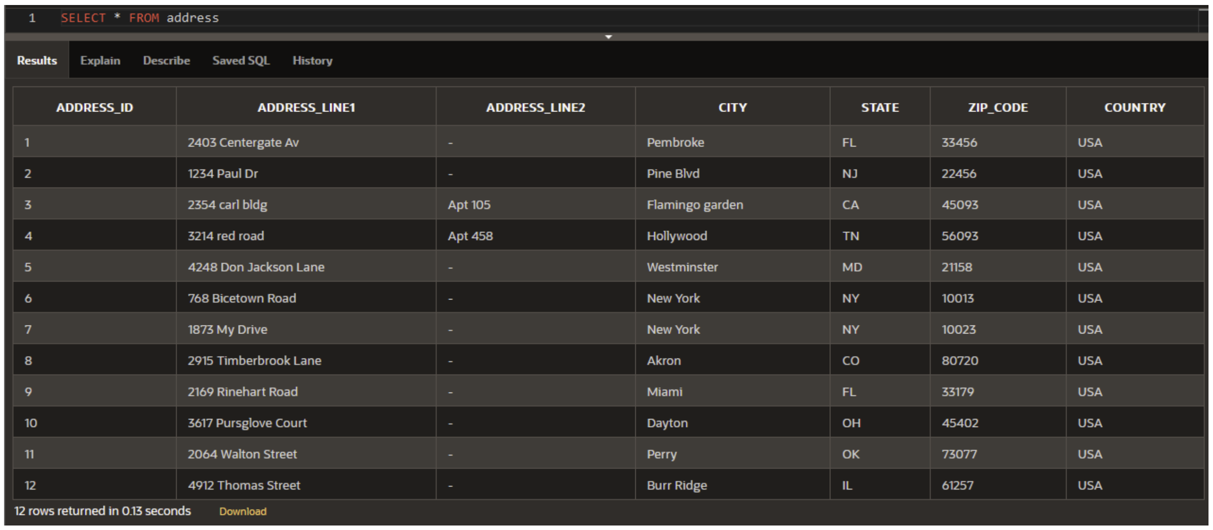
)

DDL:

```
create table address
        (address_id               int not null,
         address_line1            varchar(250) not null,
         address_line2            varchar(250) null,
         city                     varchar(100) not null,
         State           varchar(100) not null,
         zip_code                 varchar(7) not null,
         country                  varchar(100) not null,
         primary key (address_id)
         );
```

Results:

SELECT * FROM address;



```
SELECT * FROM address
```

ADDRESS_ID	ADDRESS_LINE1	ADDRESS_LINE2	CITY	STATE	ZIP_CODE	COUNTRY
1	2403 Centergate Av	-	Pembroke	FL	33456	USA
2	1234 Paul Dr	-	Pine Blvd	NJ	22456	USA
3	2354 carl bldg	Apt 105	Flamingo garden	CA	45093	USA
4	3214 red road	Apt 458	Hollywood	TN	56093	USA
5	4248 Don Jackson Lane	-	Westminster	MD	21158	USA
6	768 Bicetown Road	-	New York	NY	10013	USA
7	1873 My Drive	-	New York	NY	10023	USA
8	2915 Timberbrook Lane	-	Akron	CO	80720	USA
9	2169 Rinehart Road	-	Miami	FL	33179	USA
10	3617 Pursglove Court	-	Dayton	OH	45402	USA
11	2064 Walton Street	-	Perry	OK	73077	USA
12	4912 Thomas Street	-	Burr Ridge	IL	61257	USA

12 rows returned in 0.13 seconds

Entity: customer_address

The entity "customer_address" is the intersection table between the customer and address entities.

Normalization:

BCNF - The entity customer_address is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. It is in 3NF because there are no transitive dependencies such that A -> B where A and B are non-prime. Finally, it is in BCNF since A -> B, and A is a super key for all functional dependencies.

Schema:

customer_address

(

customer_id int,   *-- uniquely identifies which customer and part of the primary key*

address_id int,   *-- uniquely identifies which address and part of the primary key*

is_default number(1) null default 0, *-- The number that determines if an address for a customer is selected by default. Defaults to 0*

CONSTRAINT pk_customer_address_id PRIMARY KEY CLUSTERED (customer_id, address_id)

CONSTRAINT fk_customer_address_customer_id FOREIGN KEY (customer_id) REFERENCES [customer](customer_id)

CONSTRAINT fk_customer_address_address_id FOREIGN KEY (address_id) REFERENCES [address](address_id)

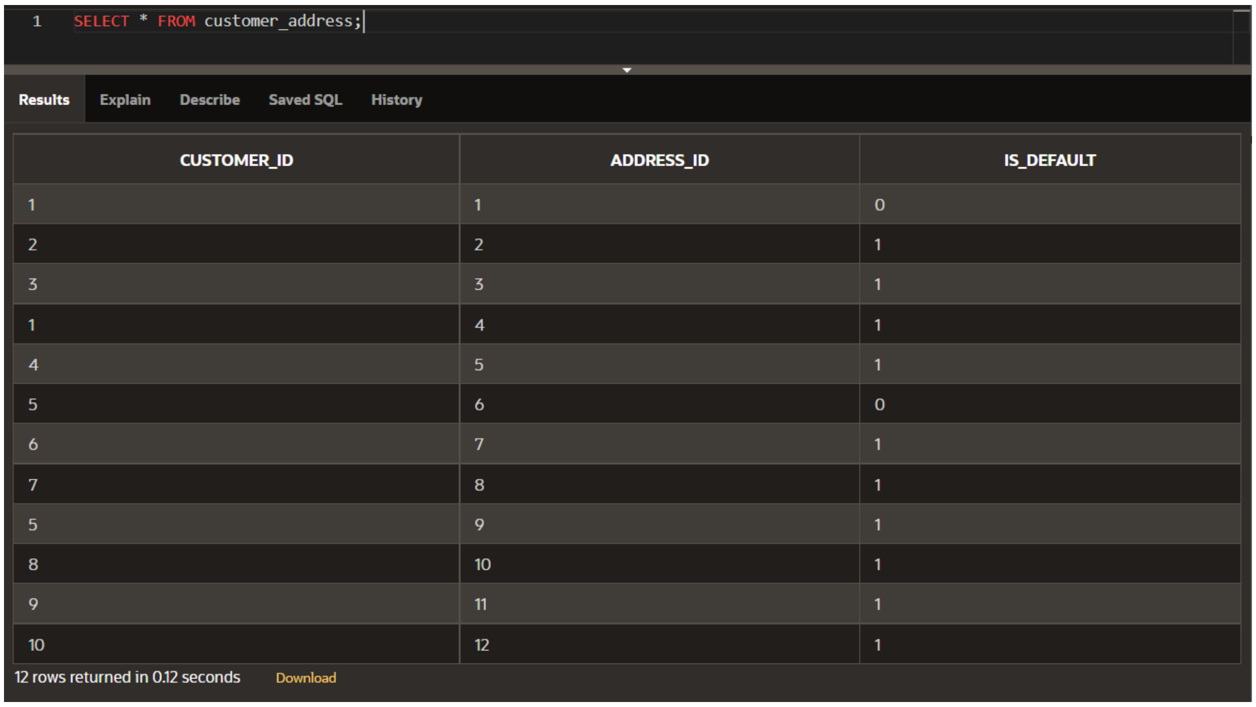
)

DDL:

```
create table customer_address
          (customer_id   int,
           address_id      int,
    is_default number(1) default 0,
           foreign key (customer_id) references customer (customer_id),
           foreign key (address_id) references address (address_id)
          );
```

Results:

SELECT * FROM customer_address;



```
SELECT * FROM customer_address;
```

CUSTOMER_ID	ADDRESS_ID	IS_DEFAULT
1	1	0
2	2	1
3	3	1
1	4	1
4	5	1
5	6	0
6	7	1
7	8	1
5	9	1
8	10	1
9	11	1
10	12	1

12 rows returned in 0.12 seconds

Entity: user_account

The entity "user_account" represents a user in the system.

Normalization:

3NF - The entity user_account is in 1NF since all attributes are single-valued, and there are no duplicate attributes or tuples. It is in 2NF because all attributes depend on the whole key. Finally, it is in 3NF because there are no transitive dependencies such that A -> B where A and B are non-prime.

Schema:

user_account

(

user_id int, *-- uniquely identifies a user*

customer_id int not null, *-- the customer is a user*

username varchar(120) not null, *-- the user's login username, which cannot be longer than 120 characters in length and is required*

password varchar(120) not null, *-- the login password (hashed and salted), which is required and has a variable length of up to 120 characters*

email varchar(120) not null, *-- the user's email address which is required and has a variable length of up to 120 characters*

CONSTRAINT pk_user_id PRIMARY KEY CLUSTERED (user_id),

CONSTRAINT fk_user_customer_id FOREIGN KEY (customer_id) REFERENCES [customer](customer_id)

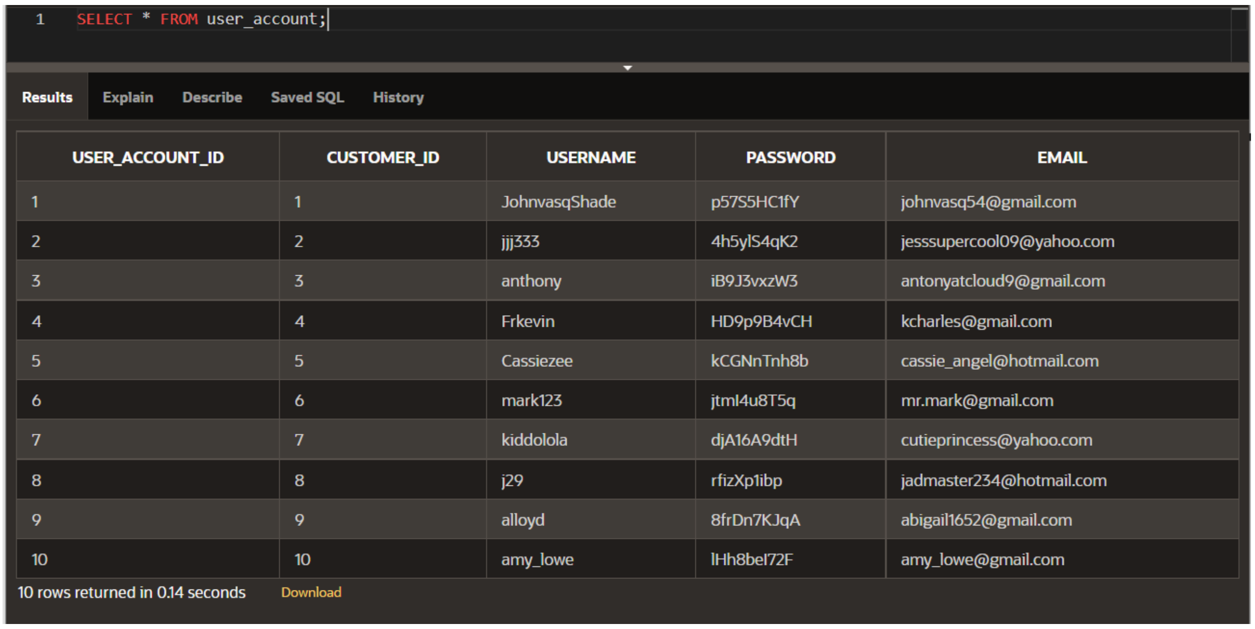
)

DDL:

```
create table user_account
        (user_account_id        int not null,
         customer_id            int not null,
         username       varchar(20) not null,
         password       varchar(20) not null,
         email          varchar(120) not null,
         primary key (user_account_id),
         foreign key (customer_id) references customer (customer_id)
        );
```

Results:

SELECT * FROM user_account;



```
SELECT * FROM user_account;
```

USER_ACCOUNT_ID	CUSTOMER_ID	USERNAME	PASSWORD	EMAIL
1	1	JohnvasqShade	p57S5HC1fY	johnvasq54@gmail.com
2	2	jjj333	4h5ylS4qK2	jesssupercool09@yahoo.com
3	3	anthony	iB9J3vxzW3	antonyatcloud9@gmail.com
4	4	Frkevin	HD9p9B4vCH	kcharles@gmail.com
5	5	Cassiezee	kCGNnTnh8b	cassie_angel@hotmail.com
6	6	mark123	jtmI4u8T5q	mr.mark@gmail.com
7	7	kiddolola	djA16A9dtH	cutieprincess@yahoo.com
8	8	j29	rfizXp1ibp	jadmaster234@hotmail.com
9	9	alloyd	8frDn7KJqA	abigail1652@gmail.com
10	10	amy_lowe	lHh8beI72F	amy_lowe@gmail.com

10 rows returned in 0.14 seconds Download

Queries

Query 1

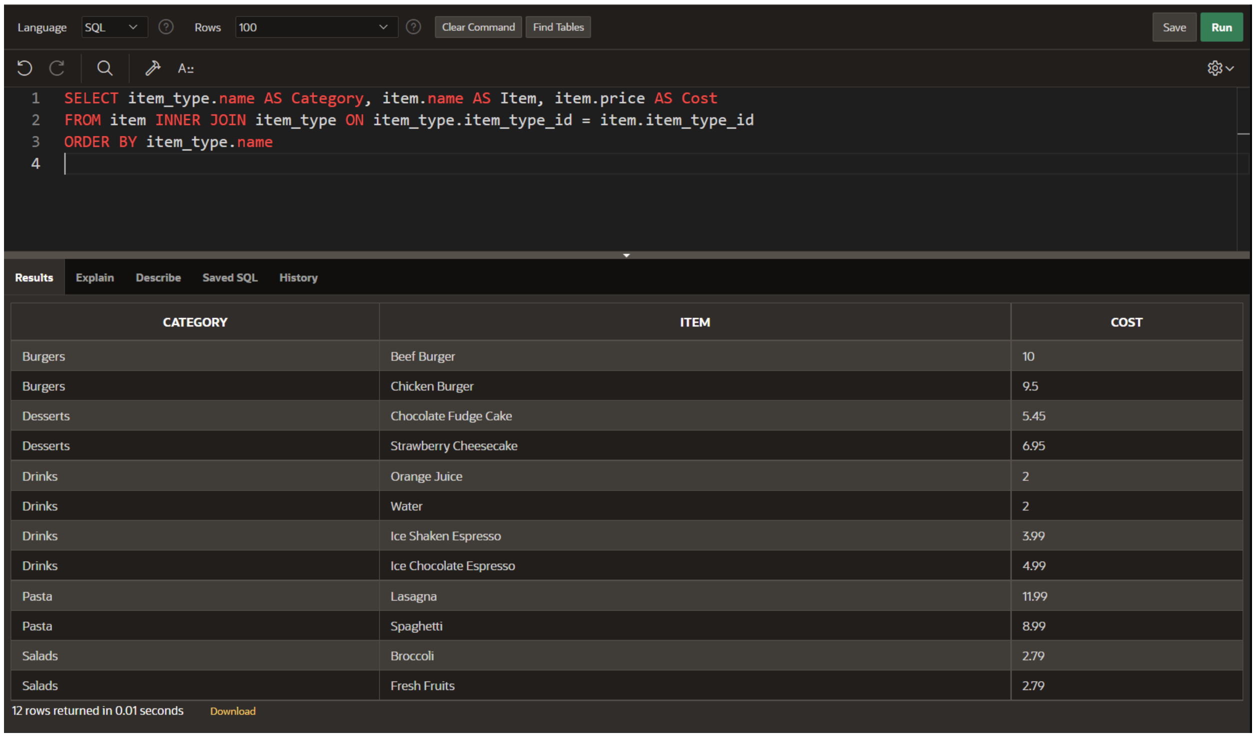
A list of all items offered, the category (item type) it belongs to, and their cost. This is useful for displaying the latest menu items to the users.

Syntax:

SELECT item_type.name AS Category, item.name AS Item, item.price AS Cost

FROM item INNER JOIN item_type ON item_type.item_type_id = item.item_type_id

ORDER BY item_type.name



```
SELECT item_type.name AS Category, item.name AS Item, item.price AS Cost
FROM item INNER JOIN item_type ON item_type.item_type_id = item.item_type_id
ORDER BY item_type.name
```

CATEGORY	ITEM	COST
Burgers	Beef Burger	10
Burgers	Chicken Burger	9.5
Desserts	Chocolate Fudge Cake	5.45
Desserts	Strawberry Cheesecake	6.95
Drinks	Orange Juice	2
Drinks	Water	2
Drinks	Ice Shaken Espresso	3.99
Drinks	Ice Chocolate Espresso	4.99
Pasta	Lasagna	11.99
Pasta	Spaghetti	8.99
Salads	Broccoli	2.79
Salads	Fresh Fruits	2.79

12 rows returned in 0.01 seconds

<u>Query 2</u>

A list of received orders (status of received) and special instructions (notes). This is useful for chefs to know which special order they should begin preparing first.

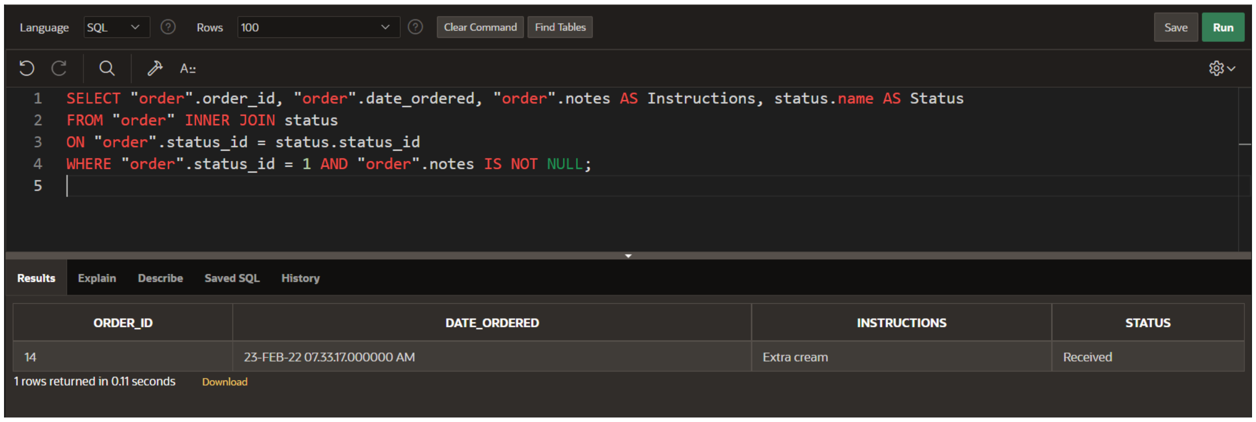
Syntax:

SELECT "order".order_id, "order".date_ordered, "order".notes AS Instructions, status.name AS Status

FROM "order" INNER JOIN status

ON "order".status_id = status.status_id

WHERE "order".status_id = 1 AND "order".notes IS NOT NULL;



```sql
SELECT "order".order_id, "order".date_ordered, "order".notes AS Instructions, status.name AS Status
FROM "order" INNER JOIN status
ON "order".status_id = status.status_id
WHERE "order".status_id = 1 AND "order".notes IS NOT NULL;
```

ORDER_ID	DATE_ORDERED	INSTRUCTIONS	STATUS
14	23-FEB-22 07.33.17.000000 AM	Extra cream	Received

1 rows returned in 0.11 seconds

<u>Query 3</u>

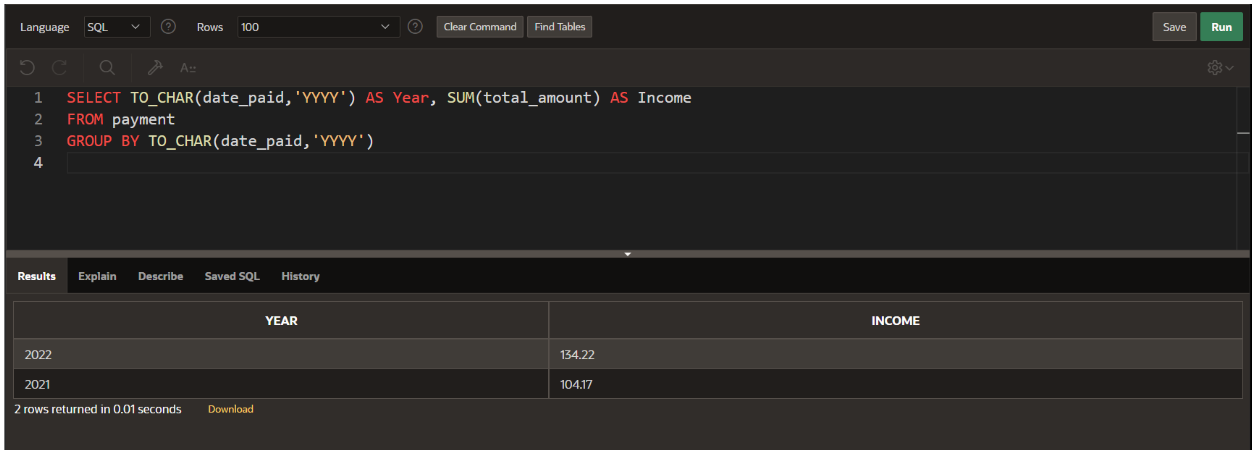
The total income made per year. This is useful for making business decisions.

Syntax:

SELECT TO_CHAR(date_paid,'YYYY') AS Year, SUM(total_amount) AS Income

FROM payment

GROUP BY TO_CHAR(date_paid,'YYYY')



```
SELECT TO_CHAR(date_paid,'YYYY') AS Year, SUM(total_amount) AS Income
FROM payment
GROUP BY TO_CHAR(date_paid,'YYYY')
```

YEAR	INCOME
2022	134.22
2021	104.17

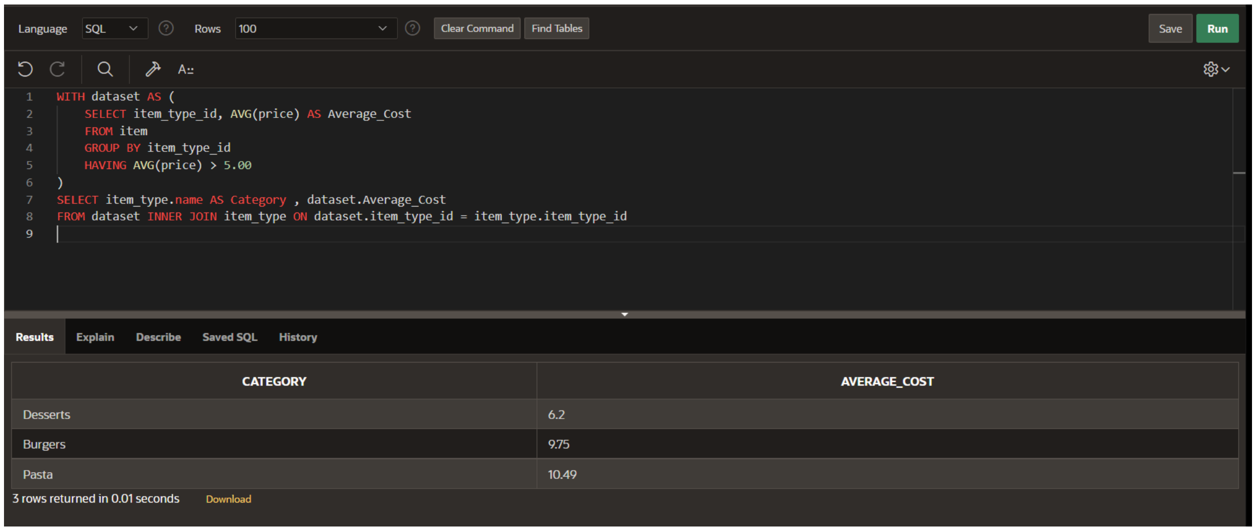
2 rows returned in 0.01 seconds

Query 4

Items types (categories) where the average cost of its items is over $5.00. This is useful for making business decisions.

Syntax:

```
WITH dataset AS (
   SELECT item_type_id, AVG(price) AS Average_Cost
   FROM item
   GROUP BY item_type_id
   HAVING AVG(price) > 5.00
)
SELECT item_type.name AS Category , dataset.Average_Cost
FROM dataset INNER JOIN item_type ON dataset.item_type_id = item_type.item_type_id
```


Language SQL Rows 100 Clear Command Find Tables Save Run

```
WITH dataset AS (
    SELECT item_type_id, AVG(price) AS Average_Cost
    FROM item
    GROUP BY item_type_id
    HAVING AVG(price) > 5.00
)
SELECT item_type.name AS Category , dataset.Average_Cost
FROM dataset INNER JOIN item_type ON dataset.item_type_id = item_type.item_type_id
```

Results Explain Describe Saved SQL History

CATEGORY	AVERAGE_COST
Desserts	6.2
Burgers	9.75
Pasta	10.49

3 rows returned in 0.01 seconds Download

Query 5

The number of orders made yearly. This is useful for making business decisions.

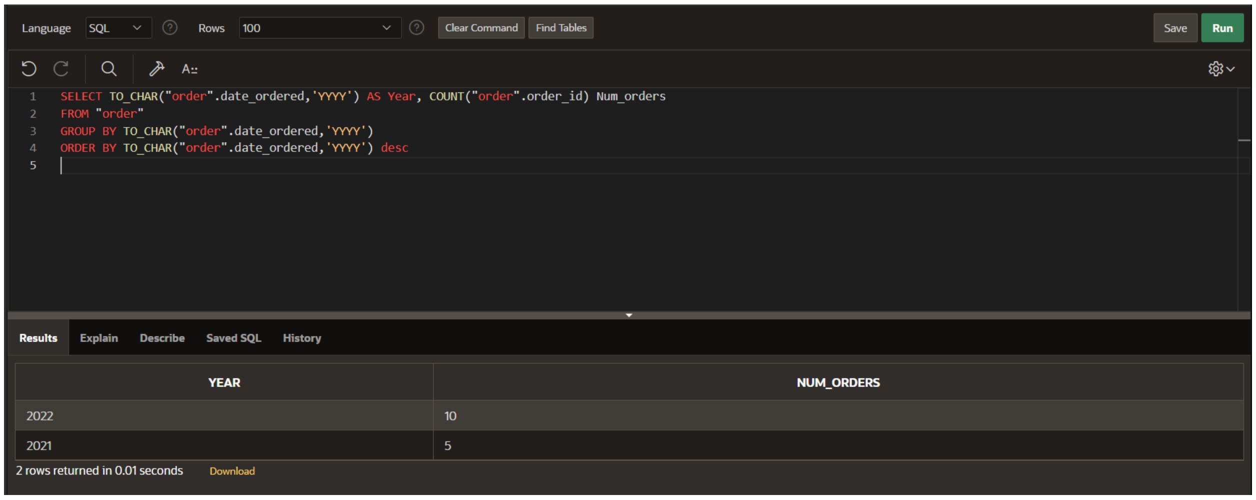
Syntax:

SELECT TO_CHAR("order".date_ordered,'YYYY') AS Year, COUNT("order".order_id) Num_orders

FROM "order"

GROUP BY TO_CHAR("order".date_ordered,'YYYY')

ORDER BY TO_CHAR("order".date_ordered,'YYYY') desc


Language SQL Rows 100 Clear Command Find Tables Save Run

```
SELECT TO_CHAR("order".date_ordered,'YYYY') AS Year, COUNT("order".order_id) Num_orders
FROM "order"
GROUP BY TO_CHAR("order".date_ordered,'YYYY')
ORDER BY TO_CHAR("order".date_ordered,'YYYY') desc
```

Results Explain Describe Saved SQL History

YEAR	NUM_ORDERS
2022	10
2021	5

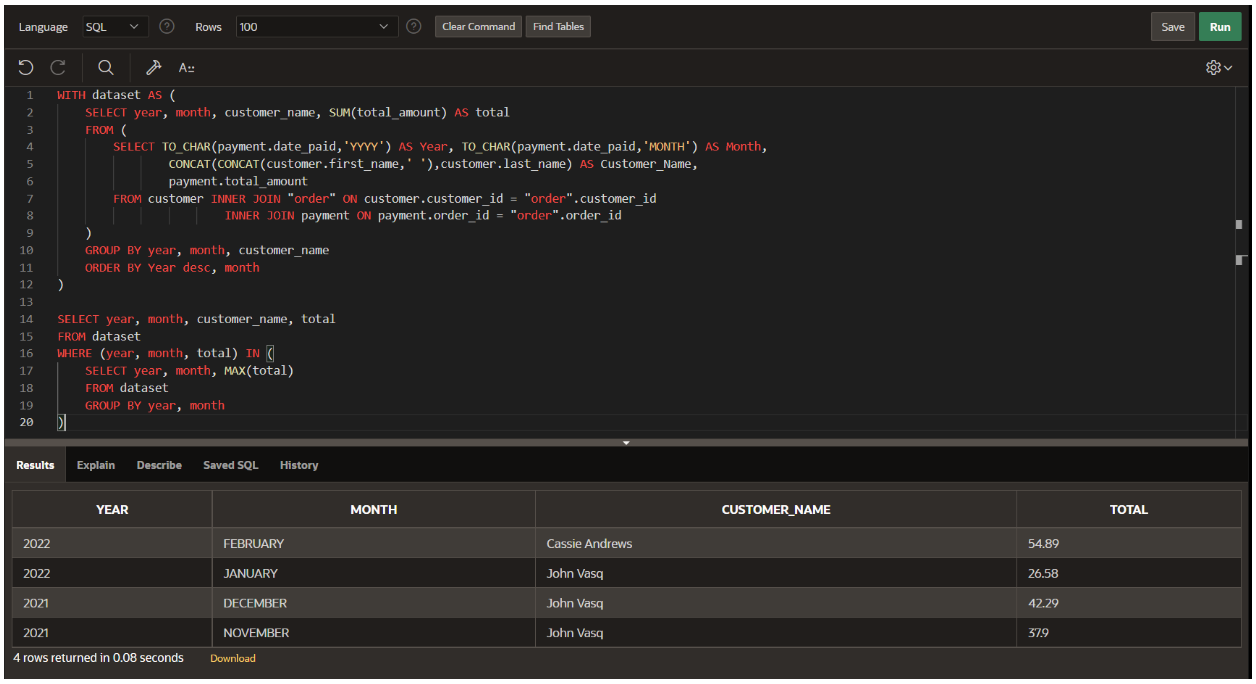
2 rows returned in 0.01 seconds Download

<u>Query 6</u>

Customer of the month! This is useful for customer appreciation day.

Syntax:

```
WITH dataset AS (
   SELECT year, month, customer_name, SUM(total_amount) AS total
   FROM (
      SELECT TO_CHAR(payment.date_paid,'YYYY') AS Year,
TO_CHAR(payment.date_paid,'MONTH') AS Month,
            CONCAT(CONCAT(customer.first_name,' '),customer.last_name) AS
Customer_Name,
            payment.total_amount
      FROM customer INNER JOIN "order" ON customer.customer_id = "order".customer_id
                  INNER JOIN payment ON payment.order_id = "order".order_id
   )
   GROUP BY year, month, customer_name
   ORDER BY Year desc, month
)

SELECT year, month, customer_name, total
FROM dataset
WHERE (year, month, total) IN (
   SELECT year, month, MAX(total)
   FROM dataset
   GROUP BY year, month
)
```



```
WITH dataset AS (
    SELECT year, month, customer_name, SUM(total_amount) AS total
    FROM (
        SELECT TO_CHAR(payment.date_paid,'YYYY') AS Year, TO_CHAR(payment.date_paid,'MONTH') AS Month,
                CONCAT(CONCAT(customer.first_name,' '),customer.last_name) AS Customer_Name,
                payment.total_amount
        FROM customer INNER JOIN "order" ON customer.customer_id = "order".customer_id
                        INNER JOIN payment ON payment.order_id = "order".order_id
    )
    GROUP BY year, month, customer_name
    ORDER BY Year desc, month
)

SELECT year, month, customer_name, total
FROM dataset
WHERE (year, month, total) IN (
    SELECT year, month, MAX(total)
    FROM dataset
    GROUP BY year, month
)
```

YEAR	MONTH	CUSTOMER_NAME	TOTAL
2022	FEBRUARY	Cassie Andrews	54.89
2022	JANUARY	John Vasq	26.58
2021	DECEMBER	John Vasq	42.29
2021	NOVEMBER	John Vasq	37.9

4 rows returned in 0.08 seconds

Query 7

Total income per month for a year. Good base for other business aggregations

Syntax:

SELECT

TO_CHAR(date_paid, 'MONTH') AS "Month",

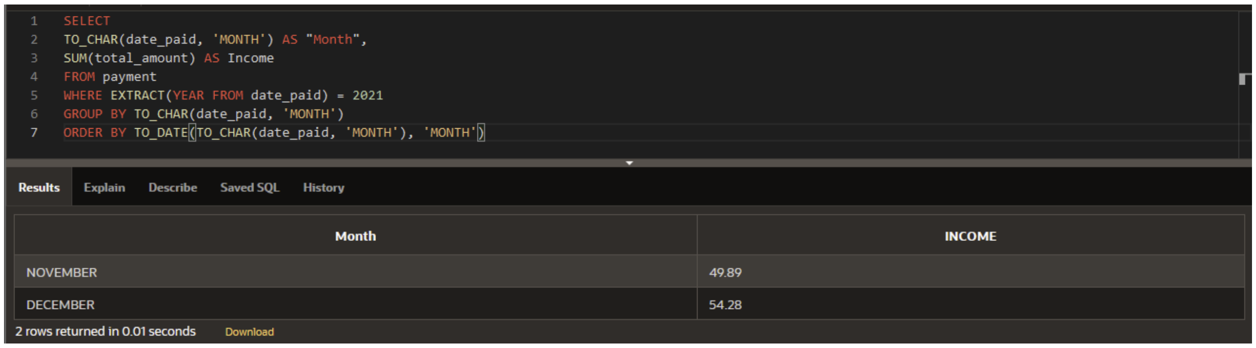
SUM(total_amount) AS Income

FROM payment

WHERE EXTRACT(YEAR FROM date_paid) = 2021

GROUP BY TO_CHAR(date_paid, 'MONTH')

ORDER BY TO_DATE(TO_CHAR(date_paid, 'MONTH'), 'MONTH')



```
SELECT
TO_CHAR(date_paid, 'MONTH') AS "Month",
SUM(total_amount) AS Income
FROM payment
WHERE EXTRACT(YEAR FROM date_paid) = 2021
GROUP BY TO_CHAR(date_paid, 'MONTH')
ORDER BY TO_DATE(TO_CHAR(date_paid, 'MONTH'), 'MONTH')
```

Month	INCOME
NOVEMBER	49.89
DECEMBER	54.28

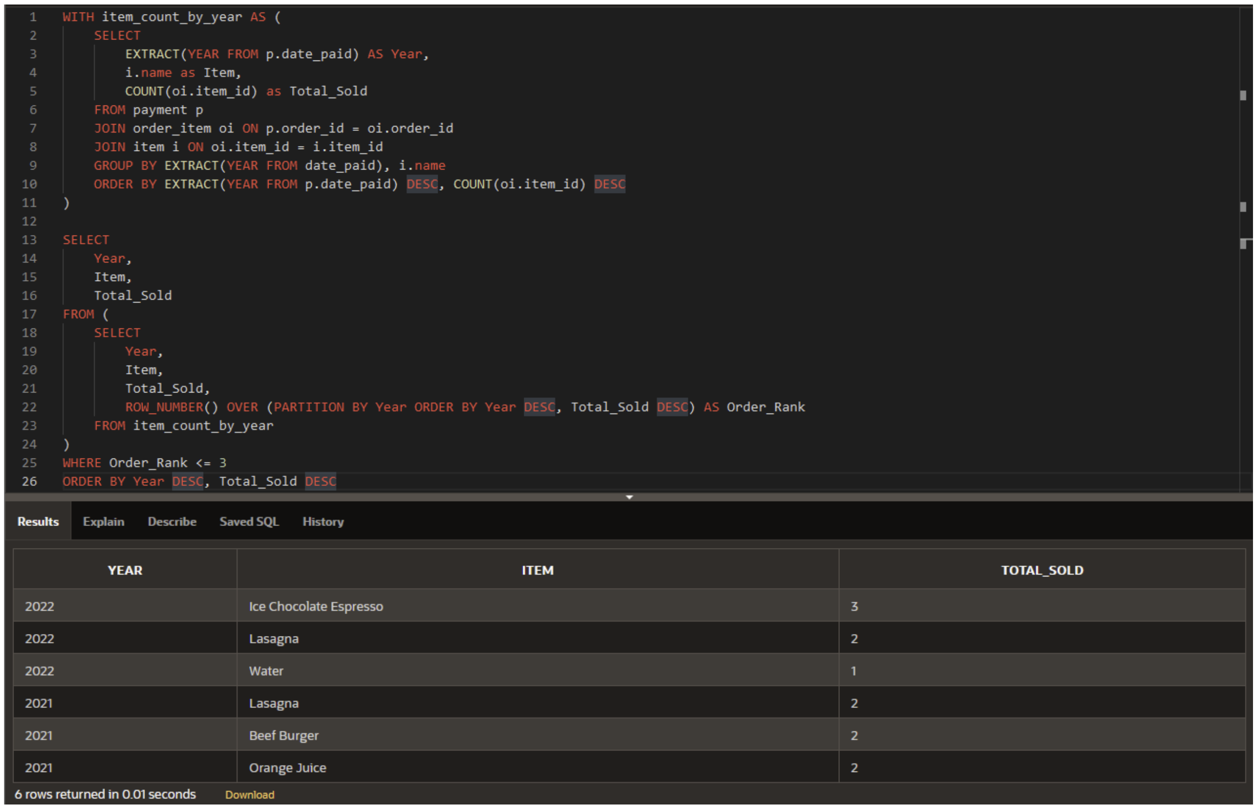
2 rows returned in 0.01 seconds

Query 8

Top 3 items sold per year. It will help in making product decisions.

Syntax:

```
WITH item_count_by_year AS (
	SELECT
	EXTRACT(YEAR FROM p.date_paid) AS Year,
	i.name as Item,
	COUNT(oi.item_id) as Total_Sold
	FROM payment p
	JOIN order_item oi ON p.order_id = oi.order_id
	JOIN item i ON oi.item_id = i.item_id
	GROUP BY EXTRACT(YEAR FROM date_paid), i.name
	ORDER BY EXTRACT(YEAR FROM p.date_paid) DESC, COUNT(oi.item_id) DESC
)
SELECT
	Year,
	Item,
	Total_Sold
FROM (
	SELECT
	Year,
	Item,
	Total_Sold,
	ROW_NUMBER() OVER (PARTITION BY Year ORDER BY Year DESC, Total_Sold
DESC) AS Order_Rank
	FROM item_count_by_year
)
WHERE Order_Rank <= 3
ORDER BY Year DESC, Total_Sold DESC
```


```
WITH item_count_by_year AS (
    SELECT
        EXTRACT(YEAR FROM p.date_paid) AS Year,
        i.name as Item,
        COUNT(oi.item_id) as Total_Sold
    FROM payment p
    JOIN order_item oi ON p.order_id = oi.order_id
    JOIN item i ON oi.item_id = i.item_id
    GROUP BY EXTRACT(YEAR FROM date_paid), i.name
    ORDER BY EXTRACT(YEAR FROM p.date_paid) DESC, COUNT(oi.item_id) DESC
)

SELECT
    Year,
    Item,
    Total_Sold
FROM (
    SELECT
        Year,
        Item,
        Total_Sold,
        ROW_NUMBER() OVER (PARTITION BY Year ORDER BY Year DESC, Total_Sold DESC) AS Order_Rank
    FROM item_count_by_year
)
WHERE Order_Rank <= 3
ORDER BY Year DESC, Total_Sold DESC
```

YEAR	ITEM	TOTAL_SOLD
2022	Ice Chocolate Espresso	3
2022	Lasagna	2
2022	Water	1
2021	Lasagna	2
2021	Beef Burger	2
2021	Orange Juice	2

6 rows returned in 0.01 seconds

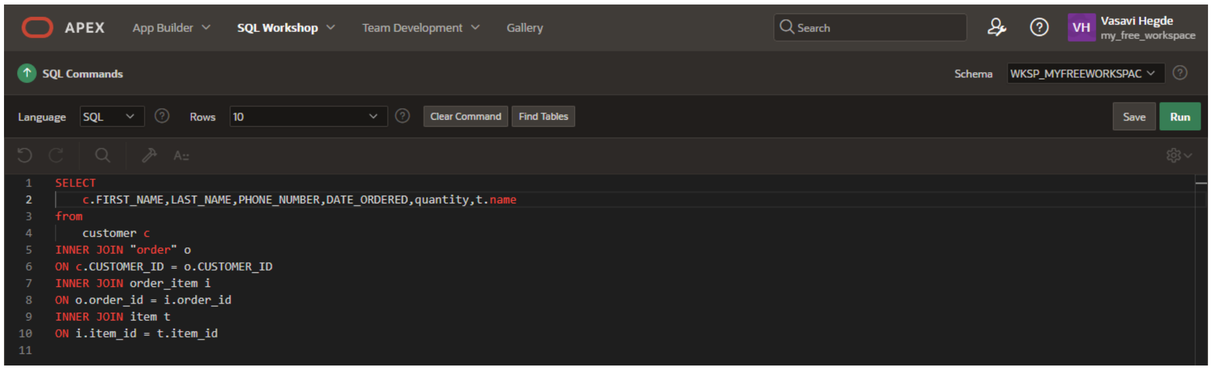
Query 9

Find the customer first name, last name, phone number, date ordered, quantity ordered, and name of the food ordered using customer, order, order_item, and item tables. This will be helpful in finding the details of customers and their food ordered.

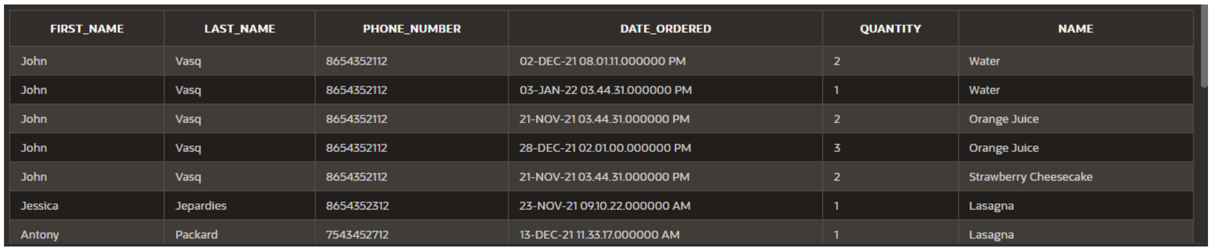
Syntax:

```
SELECT
    c.FIRST_NAME,
    LAST_NAME,
    PHONE_NUMBER,
    DATE_ORDERED,
    quantity,
    t.name
from
```

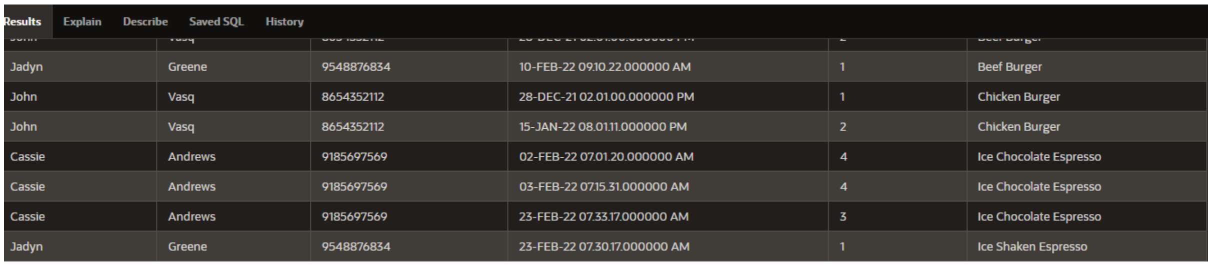
```
    customer c
INNER JOIN "order" o
ON c.CUSTOMER_ID = o.CUSTOMER_ID
INNER JOIN order_item i
ON o.order_id = i.order_id
INNER JOIN item t
ON i.item_id = t.item_id
```



```
SELECT
    c.FIRST_NAME,LAST_NAME,PHONE_NUMBER,DATE_ORDERED,quantity,t.name
from
    customer c
INNER JOIN "order" o
ON c.CUSTOMER_ID = o.CUSTOMER_ID
INNER JOIN order_item i
ON o.order_id = i.order_id
INNER JOIN item t
ON i.item_id = t.item_id
```



FIRST_NAME	LAST_NAME	PHONE_NUMBER	DATE_ORDERED	QUANTITY	NAME
John	Vasq	8654352112	02-DEC-21 08.01.11.000000 PM	2	Water
John	Vasq	8654352112	03-JAN-22 03.44.31.000000 PM	1	Water
John	Vasq	8654352112	21-NOV-21 03.44.31.000000 PM	2	Orange Juice
John	Vasq	8654352112	28-DEC-21 02.01.00.000000 PM	3	Orange Juice
John	Vasq	8654352112	21-NOV-21 03.44.31.000000 PM	2	Strawberry Cheesecake
Jessica	Jepardies	8654352312	23-NOV-21 09.10.22.000000 AM	1	Lasagna
Antony	Packard	7543452712	13-DEC-21 11.33.17.000000 AM	1	Lasagna



Jadyn	Greene	9548876834	10-FEB-22 09.10.22.000000 AM	1	Beef Burger
John	Vasq	8654352112	28-DEC-21 02.01.00.000000 PM	1	Chicken Burger
John	Vasq	8654352112	15-JAN-22 08.01.11.000000 PM	2	Chicken Burger
Cassie	Andrews	9185697569	02-FEB-22 07.01.20.000000 AM	4	Ice Chocolate Espresso
Cassie	Andrews	9185697569	03-FEB-22 07.15.31.000000 AM	4	Ice Chocolate Espresso
Cassie	Andrews	9185697569	23-FEB-22 07.33.17.000000 AM	3	Ice Chocolate Espresso
Jadyn	Greene	9548876834	23-FEB-22 07.30.17.000000 AM	1	Ice Shaken Espresso

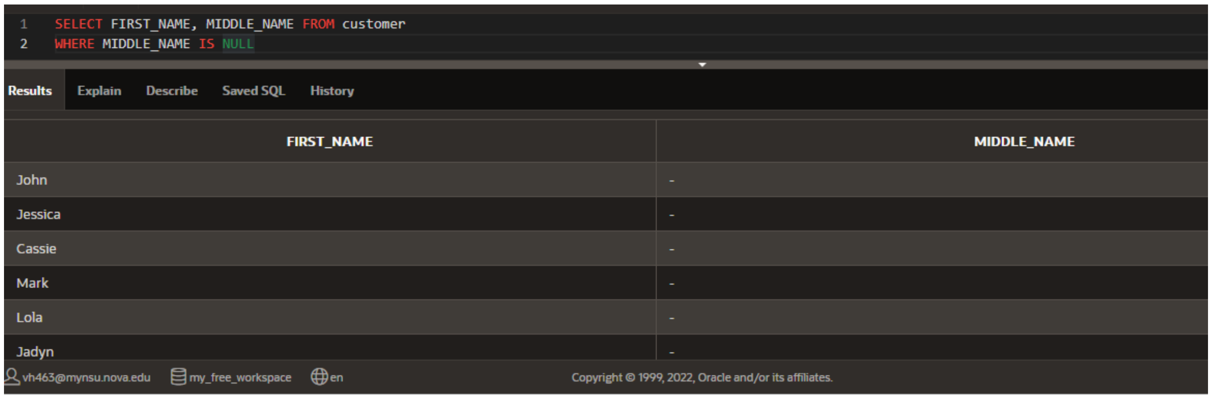
Query 10

Find the name of customers from the table whose middle name is not available. This is to find the customers name correctly as there will orders with same name.

Syntax:

SELECT FIRST_NAME, MIDDLE_NAME FROM customer

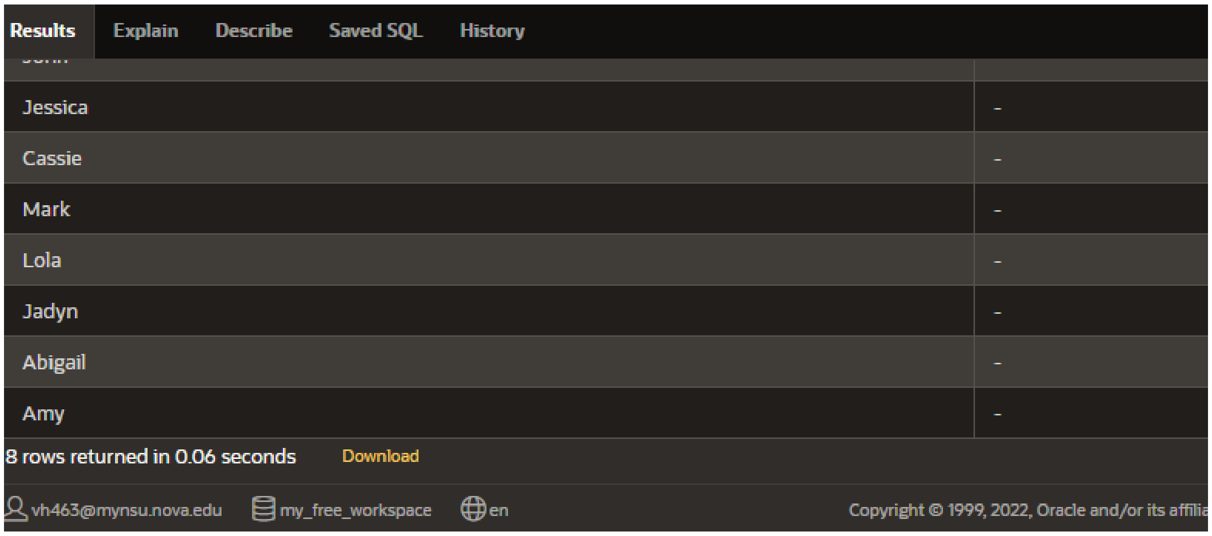
WHERE MIDDLE_NAME IS NULL



```
SELECT FIRST_NAME, MIDDLE_NAME FROM customer
WHERE MIDDLE_NAME IS NULL
```

FIRST_NAME	MIDDLE_NAME
John	-
Jessica	-
Cassie	-
Mark	-
Lola	-
Jadyn	-

vh463@mynsu.nova.edu my_free_workspace en Copyright © 1999, 2022, Oracle and/or its affiliates.



Jessica	-
Cassie	-
Mark	-
Lola	-
Jadyn	-
Abigail	-
Amy	-

8 rows returned in 0.06 seconds

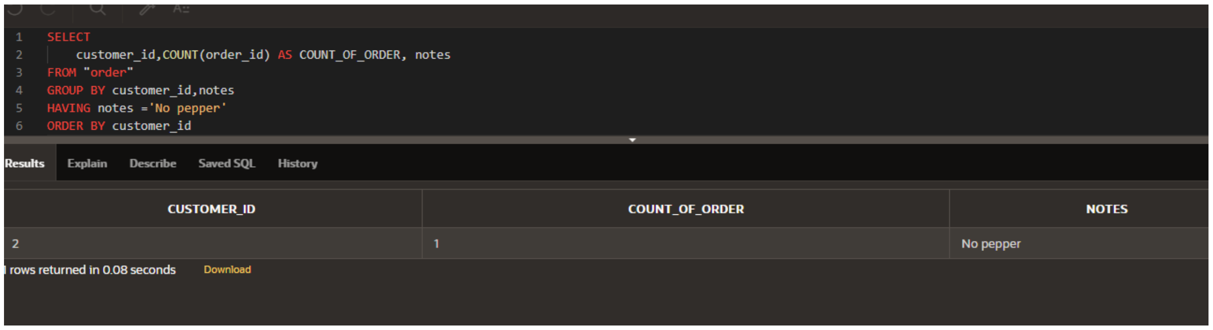
vh463@mynsu.nova.edu my_free_workspace en Copyright © 1999, 2022, Oracle and/or its affilia

Query 11

Find the customer who is having note ‘No pepper’. This is to find the food ordered with a special note.

Syntax:

```
SELECT
    customer_id,COUNT(order_id) AS COUNT_OF_ORDER, notes
FROM "order"
GROUP BY customer_id,notes
HAVING notes ='No pepper'
ORDER BY customer_id
```



```
SELECT
    customer_id,COUNT(order_id) AS COUNT_OF_ORDER, notes
FROM "order"
GROUP BY customer_id,notes
HAVING notes ='No pepper'
ORDER BY customer_id
```

CUSTOMER_ID	COUNT_OF_ORDER	NOTES
2	1	No pepper

1 rows returned in 0.08 seconds

Appendix

Drop Script

```
drop table customer_address;

drop table order_item;

drop table payment;

drop table address;

drop table user_account;

drop table item;

drop table "order";

drop table customer;

drop table delivery_method;

drop table item_type;

drop table status;

drop table payment_method;
```

Create Script

```
create table delivery_method

        (delivery_method_id     int not null,

         name                                  varchar(120) not null,

         primary key (delivery_method_id)

        );


create table item_type

        (item_type_id   int not null,

         name               varchar(120) not null,

         description   varchar(120) not null,

         primary key (item_type_id)
```

```
            );

 create table status
            (status_id int not null,
             name               varchar(120) not null,
             primary key (status_id)
            );

create table payment_method
            (payment_method_id    int not null,
             name                                   varchar(100) not null,
             primary key (payment_method_id)
            );

create table address
            (address_id                             int not null,
             address_line1                          varchar(250) not null,
             address_line2                          varchar(250) null,
             city                                   varchar(100) not null,
             State            varchar(100) not null,
             zip_code                               varchar(7) not null,
             country                                varchar(100) not null,
             primary key (address_id)
             );

create table customer
            (customer_id                int not null,
             first_name                 varchar(120) not null,
             middle_name   varchar(120) null,
             last_name                    varchar(120) not null,
             phone_numbervarchar(15) not null,
             primary key (customer_id)
            );

create table customer_address
            (customer_id               int,
             address_id                           int,
     is_default number(1) default 0,
             foreign key (customer_id) references customer (customer_id),
             foreign key (address_id) references address (address_id)
            );

create table user_account
            (user_account_id          int not null,
             customer_id               int not null,
             username                             varchar(20) not null,
             password                             varchar(20) not null,
             email                                varchar(120) not null,
             primary key (user_account_id),
             foreign key (customer_id) references customer (customer_id)
            );

create table item
            (item_id                   int not null,
     item_type_idint not null,
             name                      varchar(120) not null,
             price                     number(10,4) not null,
             primary key (item_id),
             foreign key (item_type_id) references item_type (item_type_id)
                         on delete cascade
             );



create table "order"
            (order_id int not null,
             date_ordered                timestamp not null,
             delivery_method_id      int not null,
             notes                                                   varchar(300) null,
             status_id           int not null,
             customer_id                             int not null,
             primary key (order_id),
             foreign key (delivery_method_id) references delivery_method (delivery_method_id)
                         on delete set null,
             foreign key (status_id) references status (status_id)
                         on delete set null,
             foreign key (customer_id) references customer (customer_id)
                         on delete set null
            );

create table order_item
            (order_id                               int not null,
             item_id                     int not null,
             quantity                                int not null,
             foreign key (order_id) references "order" (order_id)
                         on delete cascade,
             foreign key (item_id) references item (item_id)
                         on delete cascade
```

```
          );

create table payment
          (payment_id           int not null,
           order_id             int not null,
           total_amount   number(10,4) not null,
           date_paid            timestamp not null,
           payment_method_id    int not null,
           primary key (payment_id),
           foreign key (order_id) references "order"  (order_id)
                     on delete set null,
           foreign key (payment_method_id) references payment_method (payment_method_id)
                     on delete set null
           );
```

Insert Script

```
delete from order_item;
delete from payment;
delete from customer_address;
delete from user_account;
delete from "order";
delete from address;
delete from delivery_method;
delete from item_type;
delete from status;
delete from payment_method;
delete from item;
delete from customer;
```

```
insert into delivery_method values (1,'In-Store Pickup');

insert into delivery_method values (2,'Drop-Off Location');

insert into item_type values (1,'Drinks','Thirst quenchers! served ice cold.');

insert into item_type values (2,'Desserts','Sinfully sweet.');

insert into item_type values (3,'Pasta','Traditional itallian food.');

insert into item_type values (4,'Salads','Organically grown in our very own garden.');

insert into item_type values (5,'Burgers','Hand pressed patty served with tomatoes, lettuce and red
onion.');

insert into status values (1,'Received');

insert into status values (2,'Preparing');

insert into status values (3,'Cooking');

insert into status values (4,'Ready');

insert into status values (5,'Delivered');

insert into payment_method values (1,'Credit Card');

insert into payment_method values (2,'PayPal');

insert into item values (1,1,'Water',2);

insert into item values (2,1,'Orange Juice',2);

insert into item values (3,2,'Chocolate Fudge Cake',5.45);

insert into item values (4,2,'Strawberry Cheesecake',6.95);

insert into item values (5,3,'Lasagna',11.99);

insert into item values (6,3,'Spaghetti',8.99);

insert into item values (7,4,'Broccoli',2.79);

insert into item values (8,4,'Fresh Fruits',2.79);

insert into item values (9,5,'Beef Burger',10);

insert into item values (10,5,'Chicken Burger',9.50);

insert into item values (11,1,'Ice Chocolate Espresso',4.99);

insert into item values (12,1,'Ice Shaken Espresso',3.99);

insert into customer values (1,'John',NULL,'Vasq','8654352112');

insert into customer values (2,'Jessica',NULL,'Jepardies','8654352312');
insert into customer values (3,'Antony','Chris','Packard','7543452712');

insert into customer values (4,'Kevin','Charles','Smith','9178569856');

insert into customer values (5,'Cassie',NULL,'Andrews','9185697569');

insert into customer values (6,'Mark',NULL,'Winski','3472203720');

insert into customer values (7,'Lola',NULL,'Frey','8172420911');

insert into customer values (8,'Jadyn',NULL,'Greene','9548876834');

insert into customer values (9,'Abigail',NULL,'Lloyd','8602580399');

insert into customer values (10,'Amy',NULL,'Lowe','5408406838');

insert into address values (1,'2403 Centergate Av',NULL,'Pembroke','FL','33456','USA');

insert into address values (2,'1234 Paul Dr',NULL,'Pine Blvd','NJ','22456','USA');

insert into address values (3,'2354 carl bldg','Apt 105','Flamingo garden','CA','45093','USA');

insert into address values (4,'3214 red road','Apt 458','Hollywood','TN','56093','USA');

insert into address values (5,'4248 Don Jackson Lane',NULL,'Westminster','MD','21158','USA');

insert into address values (6,'768 Bicetown Road',NULL,'New York','NY','10013','USA');

insert into address values (7,'1873 My Drive',NULL,'New York','NY','10023','USA');

insert into address values (8,'2915 Timberbrook Lane',NULL,'Akron','CO','80720','USA');

insert into address values (9,'2169 Rinehart Road',NULL,'Miami','FL','33179','USA');

insert into address values (10,'3617 Pursglove Court',NULL,'Dayton','OH','45402','USA');

insert into address values (11,'2064 Walton Street',NULL,'Perry','OK','73077','USA');

insert into address values (12,'4912 Thomas Street',NULL,'Burr Ridge','IL','61257','USA');

insert into customer_address values (1,1,0);

insert into customer_address values (2,2,1);

insert into customer_address values (3,3,1);

insert into customer_address values (1,4,1);

insert into customer_address values (4,5,1);

insert into customer_address values (5,6,0);

insert into customer_address values (6,7,1);

insert into customer_address values (7,8,1);

insert into customer_address values (5,9,1);
insert into customer_address values (8,10,1);

insert into customer_address values (9,11,1);

insert into customer_address values (10,12,1);

insert into user_account values (1,1,'JohnvasqShade','p57S5HC1fY','johnvasq54@gmail.com');

insert into user_account values (2,2,'jjj333','4h5ylS4qK2','jesssupercool09@yahoo.com');

insert into user_account values (3,3,'anthony','iB9J3vxzW3','antonyatcloud9@gmail.com');

insert into user_account values (4,4,'Frkevin','HD9p9B4vCH','kcharles@gmail.com');

insert into user_account values (5,5,'Cassiezee','kCGNnTnh8b','cassie_angel@hotmail.com');

insert into user_account values (6,6,'mark123','jtmI4u8T5q','mr.mark@gmail.com');

insert into user_account values (7,7,'kiddolola','djA16A9dtH','cutieprincess@yahoo.com');

insert into user_account values (8,8,'j29','rfizXp1ibp','jadmaster234@hotmail.com');

insert into user_account values (9,9,'alloyd','8frDn7KJqA','abigail1652@gmail.com');

insert into user_account values (10,10,'amy_lowe','lHh8beI72F','amy_lowe@gmail.com');

insert into "order" values (1,TO_TIMESTAMP('2021-11-21 15:44:31', 'YYYY-MM-DD
HH24:MI:SS'),2,NULL,5,1);

insert into order_item values (1,2,2);

insert into order_item values (1,9,2);

insert into order_item values (1,4,2);

insert into payment values (1,1,37.9,TO_TIMESTAMP('2021-11-21 15:50:00', 'YYYY-MM-DD
HH24:MI:SS'),1);

insert into "order" values (2,TO_TIMESTAMP('2021-11-23 09:10:22', 'YYYY-MM-DD HH24:MI:SS'),2,'No
pepper please.',5,2);

insert into order_item values (2,5,1);

insert into payment values (2,2,11.99,TO_TIMESTAMP('2021-11-23 09:12:10', 'YYYY-MM-DD
HH24:MI:SS'),2);

insert into "order" values (3,TO_TIMESTAMP('2021-12-02 20:01:11', 'YYYY-MM-DD
HH24:MI:SS'),2,NULL,5,1);

insert into order_item values (3,1,2);

insert into order_item values (3,8,1);

insert into payment values (3,3,6.79,TO_TIMESTAMP('2021-12-12 20:05:15', 'YYYY-MM-DD
HH24:MI:SS'),1);
```

```
insert into "order" values (4,TO_TIMESTAMP('2021-12-13 11:33:17', 'YYYY-MM-DD
HH24:MI:SS'),1,NULL,5,3);

insert into order_item values (4,5,1);

insert into payment values (4,4,11.99,TO_TIMESTAMP('2021-12-13 11:34:10', 'YYYY-MM-DD
HH24:MI:SS'),2);

insert into "order" values (5,TO_TIMESTAMP('2021-12-28 14:01:00', 'YYYY-MM-DD
HH24:MI:SS'),2,NULL,5,1);

insert into order_item values (5,2,3);

insert into order_item values (5,9,2);

insert into order_item values (5,10,1);

insert into payment values (5,5,35.5,TO_TIMESTAMP('2021-12-28 14:15:51', 'YYYY-MM-DD
HH24:MI:SS'),2);

insert into "order" values (6,TO_TIMESTAMP('2022-01-03 15:44:31', 'YYYY-MM-DD
HH24:MI:SS'),2,NULL,5,1);

insert into order_item values (6,1,1);

insert into payment values (6,6,2,TO_TIMESTAMP('2022-01-03 15:45:10', 'YYYY-MM-DD HH24:MI:SS'),2);

insert into "order" values (7,TO_TIMESTAMP('2022-01-05 09:10:22', 'YYYY-MM-DD HH24:MI:SS'),2,'No
pepper',5,2);

insert into order_item values (7,5,1);

insert into payment values (7,7,11.99,TO_TIMESTAMP('2022-01-05 09:11:10', 'YYYY-MM-DD
HH24:MI:SS'),2);

insert into "order" values (8,TO_TIMESTAMP('2022-01-15 20:01:11', 'YYYY-MM-DD
HH24:MI:SS'),2,NULL,5,1);

insert into order_item values (8,7,2);

insert into order_item values (8,10,2);

insert into payment values (8,8,24.58,TO_TIMESTAMP('2022-01-15 20:11:23', 'YYYY-MM-DD
HH24:MI:SS'),2);

insert into "order" values (9,TO_TIMESTAMP('2022-01-25 11:33:17', 'YYYY-MM-DD
HH24:MI:SS'),1,NULL,5,3);

insert into order_item values (9,8,1);

insert into payment values (9,9,2.79,TO_TIMESTAMP('2022-01-15 11:35:41', 'YYYY-MM-DD
HH24:MI:SS'),1);
```

```
insert into "order" values (10,TO_TIMESTAMP('2022-02-02 07:01:20', 'YYYY-MM-DD
HH24:MI:SS'),1,'Extra cream',5,5);

insert into order_item values (10,11,4);

insert into payment values (10,10,19.96,TO_TIMESTAMP('2022-02-02 07:05:23', 'YYYY-MM-DD
HH24:MI:SS'),2);

insert into "order" values (11,TO_TIMESTAMP('2022-02-03 07:15:31', 'YYYY-MM-DD
HH24:MI:SS'),1,'Extra cream',5,5);

insert into order_item values (11,11,4);

insert into payment values (11,11,19.96,TO_TIMESTAMP('2022-02-03 07:18:11', 'YYYY-MM-DD
HH24:MI:SS'),2);

insert into "order" values (12,TO_TIMESTAMP('2022-02-10 09:10:22', 'YYYY-MM-DD HH24:MI:SS'),2,'Alot
of onions',5,8);

insert into order_item values (12,9,1);

insert into payment values (12,12,10,TO_TIMESTAMP('2022-02-10 09:11:56', 'YYYY-MM-DD
HH24:MI:SS'),1);

insert into "order" values (13,TO_TIMESTAMP('2022-02-23 07:30:17', 'YYYY-MM-DD
HH24:MI:SS'),1,'Extra cream and sugar',2,8);

insert into order_item values (13,12,1);

insert into payment values (13,13,3.99,TO_TIMESTAMP('2022-02-23 07:35:11', 'YYYY-MM-DD
HH24:MI:SS'),2);

insert into "order" values (14,TO_TIMESTAMP('2022-02-23 07:33:17', 'YYYY-MM-DD
HH24:MI:SS'),1,'Extra cream',1,5);

insert into order_item values (14,11,3);

insert into payment values (14,14,14.97,TO_TIMESTAMP('2022-02-23 07:40:48', 'YYYY-MM-DD
HH24:MI:SS'),2);

insert into "order" values (15,TO_TIMESTAMP('2022-02-23 07:41:25', 'YYYY-MM-DD
HH24:MI:SS'),2,NULL,1,6);

insert into order_item values (15,5,2);

insert into payment values (15,15,23.98,TO_TIMESTAMP('2022-02-23 07:50:49', 'YYYY-MM-DD
HH24:MI:SS'),2);
```