**Team Project - Chapter 6**

Normalizing the Relational Model for the Team Project and Creating a Normalized Oracle Database

**Team Four:**

* Ravi Kumar Konjeti
* Annie Keppel-Palmer
* Kishore Kanchan
* Griffin King

**Chapter 6 – Team Project: Normalizing the Relational Model for the Team Project and Creating a Normalized Oracle Database**

Read the sample project steps for this chapter and apply the same techniques to the team project that you are developing. For the team project, do the following:

**Step 6.1 - Begin with the list of the tables that the entities and relationships from the E-R diagram mapped to naturally, from the sample project section at the end of chapter 4.**

For each table on the list, identify functional dependencies and normalize the relation to BCNF. Then decide whether the resulting tables should be implemented in that form. If not, explain why.

1. **Client**

Original:

Client (clientId, firstName, lastName, street, city, state, zip, areaCode, phoneNumber)

The city and state are functionally dependant on the zip code

zip → city, state

Normalized:

Client (clientId, firstName, lastName, street, city, state, *zip*, areaCode, phoneNumber)

Zips (zip, city, state)

1. **Job**

Original:

Job (*contractNo*, type, eventName, location, *clientId , date,* time, duration, cost, *empId1, empId2,* *packageNoChosen*, totalCost, amtPaid, amtDue)

Normalized:

Job (*contractNo*, type, eventName, location, *clientId , date,* time, duration, cost, *empId1, empId2,* *packageNoChosen*, totalCost, amtPaid, amtDue)

1. **Photographer**

Original:

(empID, firstName, lastName, street, zip, areaCode, phoneNumber)

Normalization:

areaCode + telephoneNumber street, city, state, zip

Phones (areaCode, telephoneNumber, street, *zip*)

=Zips(zip, city, state)

Final Tables:

**Photographer** (empID, firstName, lastName, street, *zips*, areaCode, phoneNumber)

**Zips** (zips, city, state, zipCode)

1. **Package Menu**

Original:

(packageNo, orderNo, numWallet, num8by10, num5by7, num16by20, num11by14, price, albumCover, albumPages, albumType)

Normalization:

Already Normalized

Final Table:

**PackageMenu** (packageNo, orderNo, numWallet, num8by10, num5by7, num16by20, num11by14, price, albumCover, albumPages, albumType)

1. **Booking**

Original:

**Booking** (cliendId, duration, bookingDate, type, empId1, empId2)

Normalization:

cliendId+bookingDate →? All attributes

Same client may have two different bookings therefore an orderId can be used to make the table precise.

orderId+clientId+bookingDate → All attributes

empId1 + empId2 →? bookingDate, duration

EmpId1 and empId2 are not dependent on clientId because a client may have two bookings with different dates.

Final table:

**Booking**(orderId, clientId, bookingDate, duration, type)

**Emp\_schedule**(bookingDate, empId1, empId2)

1. **Order**

Original:

**order**(orderNo, contractNo, totalAmount, dateOrdered, packageNoOrdered)

Normalization:

There could be same package no in the table which should be normalized since all rows must me unique and there must be no duplicate row

Final Table:

**order**(orderNo, contractNo, totalAmount, dateOrdered, packageNoOrdered)

This table is already normalized.

1. **Payment**

Original:

**payment**(contractNo, amount, payType, datePaid)

Normalization:

**payment**(contractNo, amount, payType, datePaid)

This table is already normalized.

**Step 6.2 - Update the data dictionary and list of assumptions as needed.**

Current data dictionary

**amountDue:** The dollar amount that the client owes the studio

**billingHistory:** This table has entire details of the client's payment, previous services taken, photographer who attended the event.

**clientCellNumber:** The cell number of a client.

**clientCity:** The city of the mailing address of a client.

**clientEmail:** The email address of a client.

**clientFirstName:** The first name of a client.

**clientLastName:** The last (family) name of a client.

**clientName:** The first and last name of a client.

**clientState:** The state of the mailing address of a client.

**clientStreetAddress:** The street address of the mailing address of a client.

**clientZipCode:** The zip code of the mailing address of a client.

**companyName:** name of the client's company.

**contractID:** The ID number associated with a client's contract.

**dateBooked:** The date at which the session was booked

**dateOfEvent:** The date of the event that the client wishes to book.

**timeOfEvent:** The time of the event that the client wishes to book.

**depositAmount:** The dollar amount that the client deposited

**durationOfEvent:** The duration (length of time) of the event that the client wishes to book.

**eventType:** The type of event that client wishes to book.

**initialMeetingDate:** The date that the client first met with the manager.

**locationOfEvent:** The location of the event that the client wishes to book.

**numberOfAdditionalAlbums:**The number of additional albums needed (to be filled by the client)

**numberOfAdditionalPicturesNeeded:** The number of additional pictures needed (to be filled by the client)

**orderID:** Unique order ID that was given to the client while confirming the booking

**packageSelected:** The package that the client has selected for prints.

**paymentMethod:** The method of payment that the client wishes to use to pay the photography studio.

**photographerAvailability:** After entering photographer's name and date, Availability shows yes or no for the photographer's availability.

**photographerName:** The first and last name of the photographer that the client has chosen and is available on the date of the event.

**scheduleDate:** The date at which the Weekly Schedule form was started

**servicesDesired:** The types of services that the client is requesting.

**timeAvailable:** If available the desired date, timings of the availability is shown here.

Current Assumptions:

1. Each contract number will be unique even if it's for the same client.
2. A photographer can work on 3 events per day.
3. A photographer can work on one event at a time.
4. At the end of the six months, files are discarded unless the client requests additional times.
5. Once the entire payment is done, the studio gets all the amount and the photographer is paid in commission.
6. Payment for all sales is made in deposit or full at the time of contract. Payment may be by credit, cash, or check.
7. List of services provided to corporate clients must be evaluated periodically.
8. The database does not include the payment information for the photographer but only includes the payment information for the studio.
9. After the initial meeting, a client may request a booking, which is a tentative reservation for services. A confirmed reservation is made only when the client signs a contract.
10. In a package, the customer can choose different proofs for different finished pictures, but only one proof for each size.
11. The proofs to be included in an album are selected by hand in consultation with the photographer, and are not listed individually in the database. The set of proof chosen for the album will be identified as proof number 000 in the database. The layout of the pages in an album is also customized, and not stored in the database.
12. All wedding photography is scheduled for 6 hours duration, unless otherwise requested.
13. A package for passport photography will consist of two regulation-sized prints and no other pictures. It will have package number 999 in the database.
14. Albums can be produced for weddings or any other events, but not for portraits.
15. The total package for weddings or other events can include both a package of various-sized pictures and an album.
16. After the initial meeting, a client may request a booking, which is a tentative reservation for services. A confirmed reservation is made only when the client signs a contract. (Note: This assumption was used for the relational model, but dropped for the object-relational one.)
17. In a package, the customer can choose different proofs for different finished pictures, but only one proof for each size.
18. The proofs to be included in an album are selected by hand in consultation with the photographer, and are not listed individually in the database. The set of proof chosen for the album will be identified as proof number 000 in the database. The layout of the pages in an album is also customized, and not stored in the database.
19. All wedding photography is scheduled for 6 hours duration, unless otherwise requested.
20. A package for passport photography will consist of two regulation-sized prints and no other pictures. It will have package number 999 in the database.
21. Albums can be produced for weddings or any other events, but not for portraits.
22. The total package for weddings or other events can include both a package of various-sized pictures and an album.

**Step 6.3 - For each table, write the table name and write out the names, data types, and sizes of all the data items.**

Identify any constraints, using the conventions of the DBMS you will use for implementation.

Table Zips

zip CHAR 5

city VARCHAR2 15

state CHAR 2

TABLE Client

clientId CHAR 10

firstName VARCHAR2 15

lastName VARCHAR2 20

street VARCHAR2 50

zip CHAR 5

areaCode CHAR 3

phoneNumber CHAR 7

TABLE Job

contractNo CHAR 10

type VARCHAR2 15

eventName VARCHAR2 20

location VARCHAR2 20

date DATE

clientId CHAR 10

duration CHAR 100

time CHAR 100

empId1 CHAR 10

empId2 CHAR 10

cost NUMBER 8,2

packageNoChosen CHAR 100

totalCost NUMBER 8,2

amtDate DATE

amtPaid NUMBER 8,2

amtDue NUMBER 8,2

TABLE Photographer

empID CHAR 10

firstName VARCHAR2 15

lastName VARCHAR2 20

areaCode CHAR 3

phoneNumber CHAR 7

TABLE Zip

City VARCHAR2 15

State CHAR 2

Zip CHAR 5

TABLE Package Menu

packageNo CHAR 10

orderNo NUMBER 6

Price NUMBER 4

numWallet CHAR 4

num8by10 CHAR 4

num5by7 CHAR 4

num16by20 CHAR 4

num11by14 CHAR 4

albumCover CHAR 3

albumPages CHAR 3

albumType CHAR 8

TABLE Booking

orderId NUMBER 10

clientId CHAR 10

duration NUMBER 8,2  
type VARCHAR2 10

TABLE Emp\_schhedule

bookingDate DATE

empId2 CHAR 10

empId1 CHAR 10

TABLE Payment

ContractNo NUMBER 8,2

DatePaid DATE

Paytype CHAR 20

Amount NUMBER 8,2

TABLE Order

orderNo NUMBER 6

contractNo CHAR 10

totalAmount NUMBER 8,2

dateOrdered DATE

packageNoOrdered CHAR 10

**Step 6.4 - Write and execute SQL statements to create all the tables needed to implement the design.**

CREATE TABLE Client (

clientId CHAR(10),

firstName VARCHAR2(15),

lastName VARCHAR2(20),

street VARCHAR2(50),

zip CHAR(5),

areaCode CHAR(3),

phoneNumber CHAR(7),

CONSTRAINT Client\_clientId\_pk PRIMARY KEY (clientId),

CONSTRAINT Client\_zip\_fk FOREIGN KEY (zip) REFERENCES Zips (zip));

CREATE TABLE Zips (

zip CHAR(5),

city VARCHAR2(15),

state CHAR(2),

CONSTRAINT Zips\_zip\_pk PRIMARY KEY (zip));

CREATE TABLE Job (

contractNo CHAR(10),

type VARCHAR2(20),

eventName VARCHAR2(25),

location VARCHAR2(30),

dateHired DATE,

clientId CHAR(10),

duration CHAR(3),

time CHAR(4),

empId1 CHAR(10),

empId2 CHAR(10),

cost NUMBER(8,2),

packageNoOrdered CHAR(100),

totalCost NUMBER(8,2),

amtDate DATE,

amtPaid NUMBER(8,2),

amtDue NUMBER(8,2),

CONSTRAINT Jobs\_contractNo\_pk PRIMARY KEY (contractNo),

CONSTRAINT Job\_contractNo\_fk FOREIGN KEY(contractNo) REFERENCES Proof (proofNo),

CONSTRAINT Job\_packageNoOrdered\_fk FOREIGN KEY(packageNoOrdered) REFERENCES OrderInfo (packageNoOrdered));

CREATE TABLE PackageMenu (

orderNo NUMBER(6),

packageNo CHAR(10),

Price NUMBER(4),

numWallet CHAR(4),

num8by10 CHAR(4),

num5by7 CHAR(4),

num16by20 CHAR(4),

num11by14 CHAR(4),

albumCover CHAR(3),

albumPages CHAR(3),

albumType CHAR(8),

CONSTRAINT packageMenu\_packageNo\_pk PRIMARY KEY (packageNo),

CONSTRAINT packageMenu\_packageNo\_fk FOREIGN KEY (packageNo) REFERENCES OrderInfo (orderId));

CREATE TABLE Photographer (

empID CHAR(10),

firstName VARCHAR2(15),

lastName VARCHAR2(20),

areaCode CHAR(3),

phoneNumber CHAR(7),

street VARCHAR2(50),

zip CHAR(5),

CONSTRAINT Photographer\_empID\_pk PRIMARY KEY (empID),

CONSTRAINT Photographer\_zips\_fk FOREIGN KEY (zip) REFERENCES Zips(zip));

CREATE TABLE Booking(

orderId NUMBER(8,2),

clientId CHAR(10),

duration NUMBER(8,2),

bookingDate DATE,

type VARCHAR2(10),

CONSTRAINT Booking\_clientId\_orderId\_pk PRIMARY KEY (orderId));

CREATE TABLE Emp\_schedule(

bookingDate DATE,

empId1 CHAR(10),

empId2 CHAR(10),

CONSTRAINT EmpSched\_bookingDate\_pk PRIMARY KEY (bookingDate));

CREATE TABLE OrderInfo(

orderNo NUMBER(6),

contractNo CHAR(10),

totalAmount NUMBER(8,2),

dateOrdered DATE,

packageNoOrdered CHAR(100),

CONSTRAINT OrderInfo\_packageNoOrdered\_pk PRIMARY KEY (packageNoOrdered),

CONSTRAINT OrderInfo\_orderNo\_pk UNIQUE (orderNo));

CREATE TABLE Payment(

contractNo CHAR(10),

amount NUMBER(8,2),

payType VARCHAR2(15),

datePaid DATE,

CONSTRAINT Payment\_contractNo\_pk PRIMARY KEY (contractNo));

CREATE TABLE Meeting(

clientId CHAR(10),

meetDate DATE,

meetTime CHAR(10),

repName VARCHAR(25),

CONSTRAINT Meeting\_clientId\_pk PRIMARY KEY (clientId));

CREATE TABLE OrderItem(

orderNo CHAR(10),

proofNo CHAR(10),

dateDelivered DATE,

quantity CHAR(10),

sizeOfOrder VARCHAR2(20),

CONSTRAINT OrderItem\_orderNo\_pk PRIMARY KEY (orderNo));

CREATE TABLE Proof(

proofNo CHAR(10),

contractNo CHAR(10),

quality VARCHAR(20),

CONSTRAINT Proof\_proofNo\_pk PRIMARY KEY (proofNo),

CONSTRAINT Proof\_contractNo\_pk UNIQUE (contractNo));

**Step 6.5 - Write and execute SQL statements to create indexes for foreign keys and any other columns that will be used most often for queries.**

CREATE INDEX Client\_zip ON Client(zip);

CREATE INDEX Job\_contractNo ON Job(contractNo);

CREATE INDEX Job\_packageNoChosen ON Job(packageNoOrdered);

CREATE INDEX packageMenu\_packageNo ON packageMenu(packageNo);

CREATE INDEX Photographer\_zips ON Photographer(zip);

**Step 6.6 - Write and execute SQL statements to insert at least five records in each table, preserving all constraints.**

Put in enough data to demonstrate how the database will function.

Find jobs of people who work either as an sales consultant or event manager

SELECT type

FROM Job

WHERE type = 'sales consultant' OR type = 'event manager';

Find Client ID’s of people in meetings whose names are not Dinkleberg.

SELECT clientId

FROM Meeting

WHERE repName != 'Dinkleberg';

Find the Employee IDs of all photographers contracted by the company that have an area code of 203.

SELECT empID

FROM photographer

WHERE areaCode = '914';

Find the Client ID’s of people who made bookings between the month of November and December.

SELECT clientId

FROM Booking

WHERE bookingDate >= date '2021-11-01' AND bookingDate <= date '2021-12-31';

Find order numbers of orders that are less than $1000.

SELECT orderNo

FROM OrderInfo

WHERE totalAmount < '1000';

**Step 6.7 - Write and execute SQL statements that will process five non-routine requests for information from the database just created.**

Client:

INSERT INTO Client VALUES('0174927594','Branislav','Herleifr','Garfield Land','06880','203','4375080');

INSERT INTO Client VALUES('9380410983','Apollonius','Martina','John Ave','08332','223','1032633');

INSERT INTO Client VALUES('2617920394','Mavuto','Luisito','George Pointe','48236','204','2130478');

INSERT INTO Client VALUES('1029831029','Gertruda','Brigitta','Rock Creek Rd','59715','294','6593932');

INSERT INTO Client VALUES('6839230293','Agapito','Loida','Olive St','11374','987','7171934');

Zips:

INSERT INTO Zips VALUES('06880','Westport','CT');

INSERT INTO Zips VALUES('08332','Millville','NJ');

INSERT INTO Zips VALUES('48236','Grosse Pointe','MI');

INSERT INTO Zips VALUES('59715','Bozeman','MT');

INSERT INTO Zips VALUES('11374','Rego Park','NY');

Job:

INSERT INTO Job VALUES('8203749204','photographer','Philly Photowalk','Philadelphia Museum of Art','23-Oct-2021','8295629475','180','1200','8651474881','4678017390','1400','6456389522','1500','15-Nov-2022','700','300');

INSERT INTO Job VALUES('0293420923','studio assistant','Fall Mini-Sessions','Gifford Pinchot State Park','18-Oct-2021','0365025407','210','0900','8684827251','9527539711','600','9128742031','650','04-Jun-2022','100','400');

INSERT INTO Job VALUES('1029482394','event manager','Photography is Medicine','Cafe Lift','23-Oct-2021','2710462947','270','1000','8450197012','3438273057','100','5778796063','120','17-Feb-2022','0','0');

INSERT INTO Job VALUES('6590928341','equipment technician','Virtual Photo Club','United States','28-Oct-2021','1947593720','030','0400','1962147339','4569320979','500','9733692355','550','19-May-2022','250','500');

INSERT INTO Job VALUES('9857234213','sales consultant','Abstracts in Nature','Huntington Memorial Library','23-Oct-2021','2310897436','120','0600','6539512487','5657028875','3000','4506786445','3300','29-Jul-2022','0','700');

Booking:

INSERT INTO Booking VALUES('341135','0174927594','7',date '2021-10-23','?type');

INSERT INTO Booking VALUES('343636','0174927599','2',date '2021-10-29','?type');

INSERT INTO Booking VALUES('3422337','0174927605','10',date '2021-11-01','?type');

INSERT INTO Booking VALUES('344338','0174927635','5',date '2021-11-05','?type');

INSERT INTO Booking VALUES('345339','0174927655','4',date '2021-12-12','?type');

Employee Schedule:

INSERT INTO Emp\_Schedule VALUES(date '2021-10-23', '001', '010');

INSERT INTO Emp\_Schedule VALUES(date '2021-10-29', '012', '001');

INSERT INTO Emp\_Schedule VALUES(date '2021-11-01', '014', '015');

INSERT INTO Emp\_Schedule VALUES(date '2021-11-05', '011', '010');

INSERT INTO Emp\_Schedule VALUES(date '2021-12-12', '011', '012');

Order Info:

INSERT INTO OrderInfo VALUES('343535','8203749204','12301',date '2021-10-23','6456389522');

INSERT INTO OrderInfo VALUES('626595','7213789298','14382',date '2021-10-29','6456389523');

INSERT INTO OrderInfo VALUES('94335','8207749904','10330',date '2021-11-01','6456389524');

INSERT INTO OrderInfo VALUES('15935','6703741199','12666',date '2021-11-05','6456389529');

INSERT INTO OrderInfo VALUES('393535','8203746204','15309',date '2021-11-12','6456389562');

Payment:

INSERT INTO Payment VALUES('343335','12500','Credit Card',date '2021-09-11');

INSERT INTO Payment VALUES('343133','11200','Cash',date '2021-09-11');

INSERT INTO Payment VALUES('243338','12550','Debit Card',date '2021-09-16');

INSERT INTO Payment VALUES('343335','11500','Credit Card',date '2021-09-11');

INSERT INTO Payment VALUES('345332','10550','Credit Card',date '2021-09-19');

Photographer:

INSERT INTO Photographer VALUES('1234567890', 'Annie', 'Palmer', '917', '3849587','Garfield Land','06880');

INSERT INTO Photographer VALUES('5830593789', 'Kishore', 'Smith', '914', '4830593','John Ave','08332');

INSERT INTO Photographer VALUES('3820584930', 'Griffin', 'James', '495', '4058493','George Pointe','48236');

INSERT INTO Photographer VALUES('5940385039', 'Ravi', 'Jones', '345', '5869403','Rock Creek Rd','59715');

INSERT INTO Photographer VALUES('1583937290', 'James', 'Brown', '123', '4567889','Olive St','11374');

Package Menu:

INSERT INTO PackageMenu VALUES('123456', '1849738593', '455', '1', '2', '3', '4', '5', 'YES', '16', 'BOOK');

INSERT INTO PackageMenu VALUES('038572', '4820583759', '489', '2', '4', '56', '76', '32', 'YES', '171', 'ALBUM');

INSERT INTO PackageMenu VALUES('340829', '4825724578', '299', '59', '1', '34', '56', '5', 'NO', '155', 'BOOK');

INSERT INTO PackageMenu VALUES('482058', '6749302857', '1094', '90', '67', '4', '7', '8', 'NO', '168', 'LOOSE');

INSERT INTO PackageMenu VALUES('248503', '5830583789', '4827', '48', '56', '12', '10', '11', 'YES', '138', 'BOOK');

Meeting:

INSERT INTO Meeting VALUES('0174927594',date '2021-10-23','22:22:34','CHARLIE');

INSERT INTO Meeting VALUES('0174927599',date '2021-10-29','22:22:34','SCOTT');

INSERT INTO Meeting VALUES('0174927605',date '2021-11-01','22:22:34','ZAHID');

INSERT INTO Meeting VALUES('0174927635',date '2021-11-05','22:22:34','TRAVIS');

INSERT INTO Meeting VALUES('0174927634',date '2021-11-04','22:22:34','BONKERS');

Proof:

INSERT INTO Proof VALUES('434353','0174927','RAW');

INSERT INTO Proof VALUES('353563','0174927','RAW');

INSERT INTO Proof VALUES('356664','0174927','RAW');

INSERT INTO Proof VALUES('353646','0174927','RAW');

INSERT INTO Proof VALUES('353645','0174927','RAW');

Order Item:

INSERT INTO OrderItem VALUES('343535','353353',date '2020-4-22','34','09345709');

INSERT INTO OrderItem VALUES('343567','353535',date '2020-4-24','34','32454349');

INSERT INTO OrderItem VALUES('343534','353645',date '2020-5-22','34','23029434');

INSERT INTO OrderItem VALUES('343578','353535',date '2020-5-25','34','30249345');

INSERT INTO OrderItem VALUES('343588','353533',date '2020-6-30','34','10923010');

**Step 6.8 - Write and execute SQL statements to create at least one trigger.**

CREATE TRIGGER number\_of\_album\_pages

before UPDATE

ON PackageMenu

FOR EACH ROW

BEGIN

UPDATE PackageMenu

SET albumPages = numWallet + Num8by10 + Num5by7 + Num16by20 + Num11by14;

END

**Step 6.9 - Write and execute SQL statements to demonstrate that the trigger is working as expected.**

To demonstrate that the trigger is working as expected, provide a screenshot of the data before and after the trigger is executed.

UPDATE PackageMenu

SET num8by10 = 3

WHERE PackageNo = 1849738593;

UPDATE PackageMenu

SET num5by7 = 1

WHERE PackageNo = 6749302857;

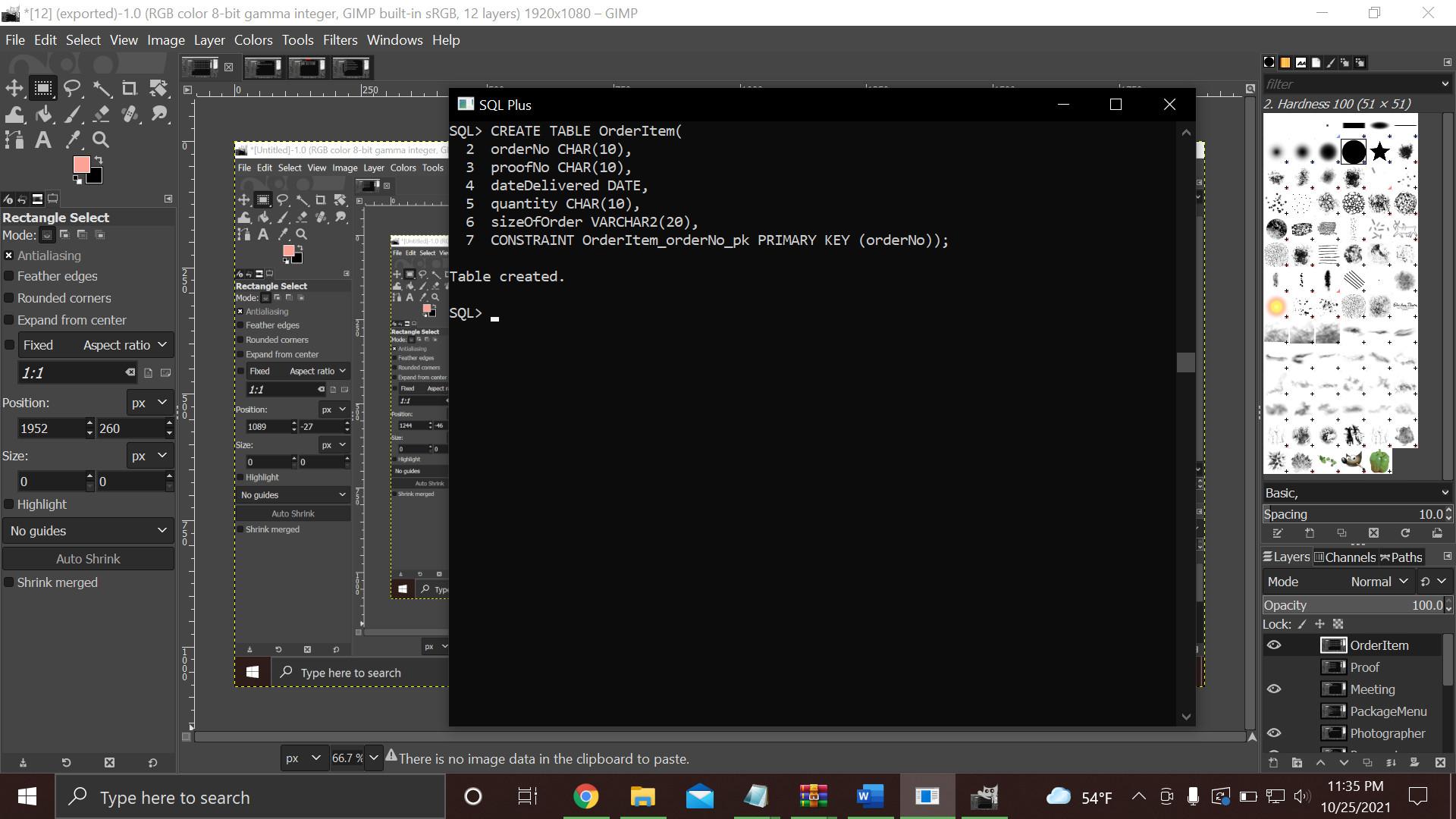
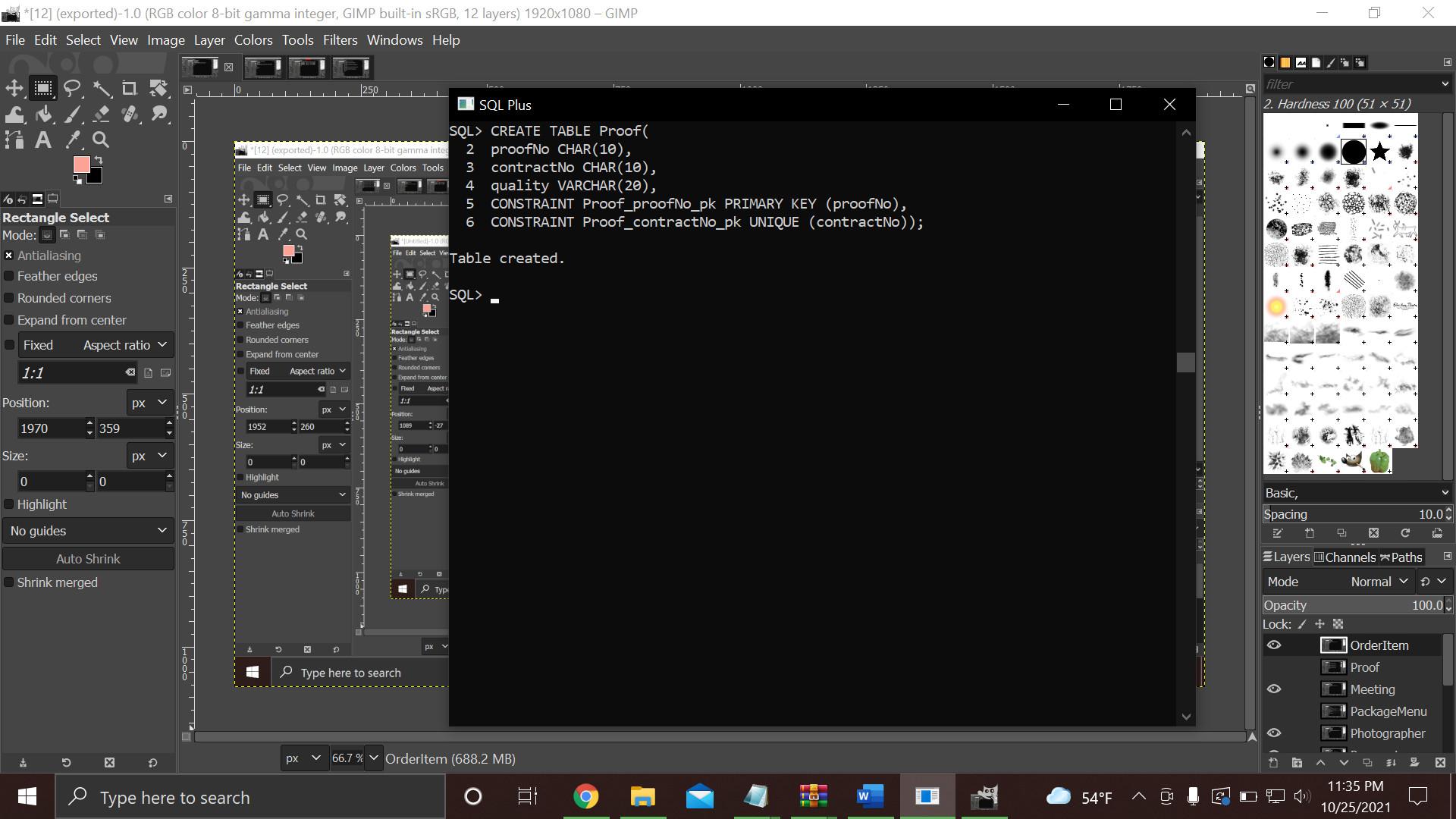
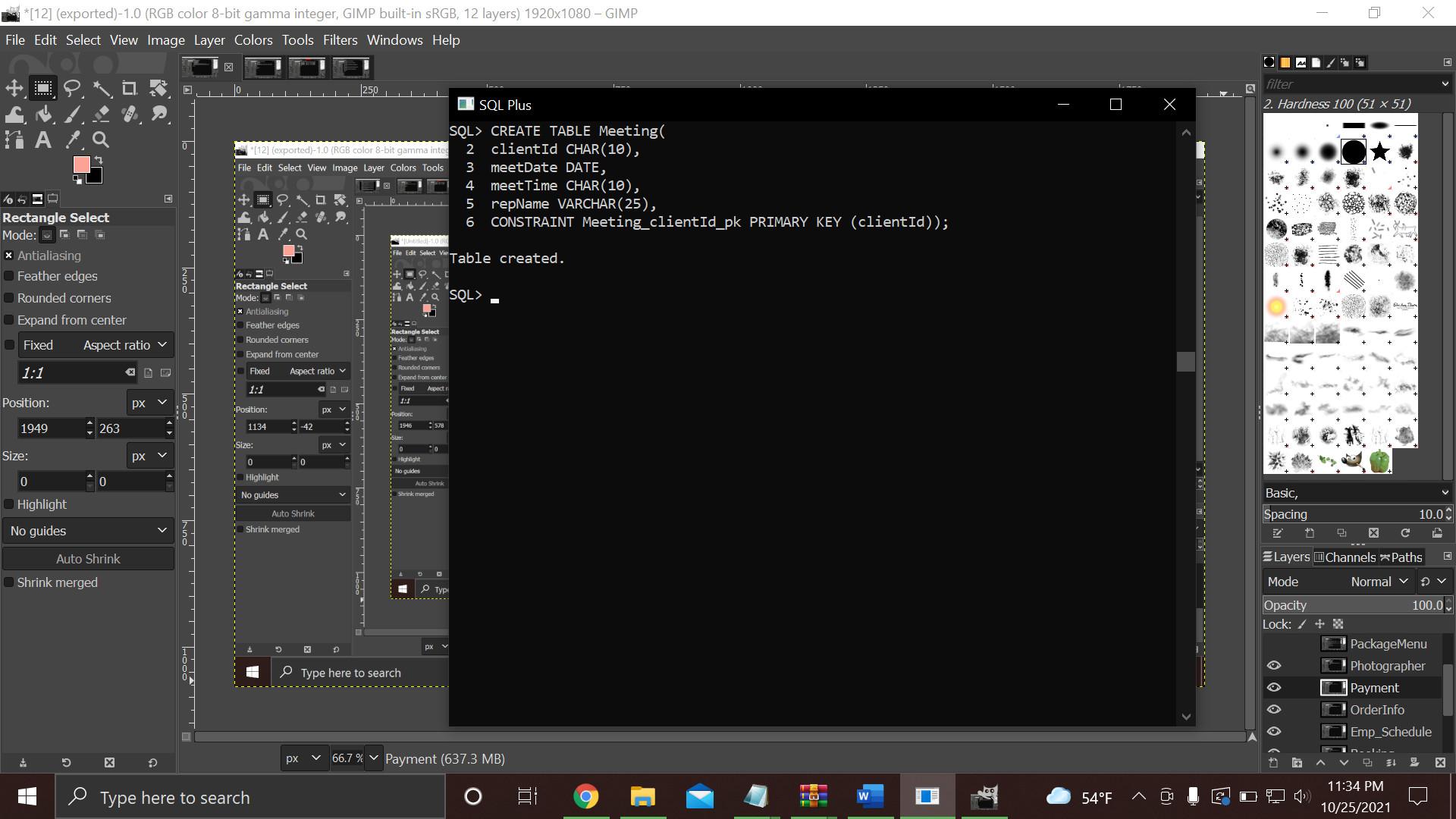
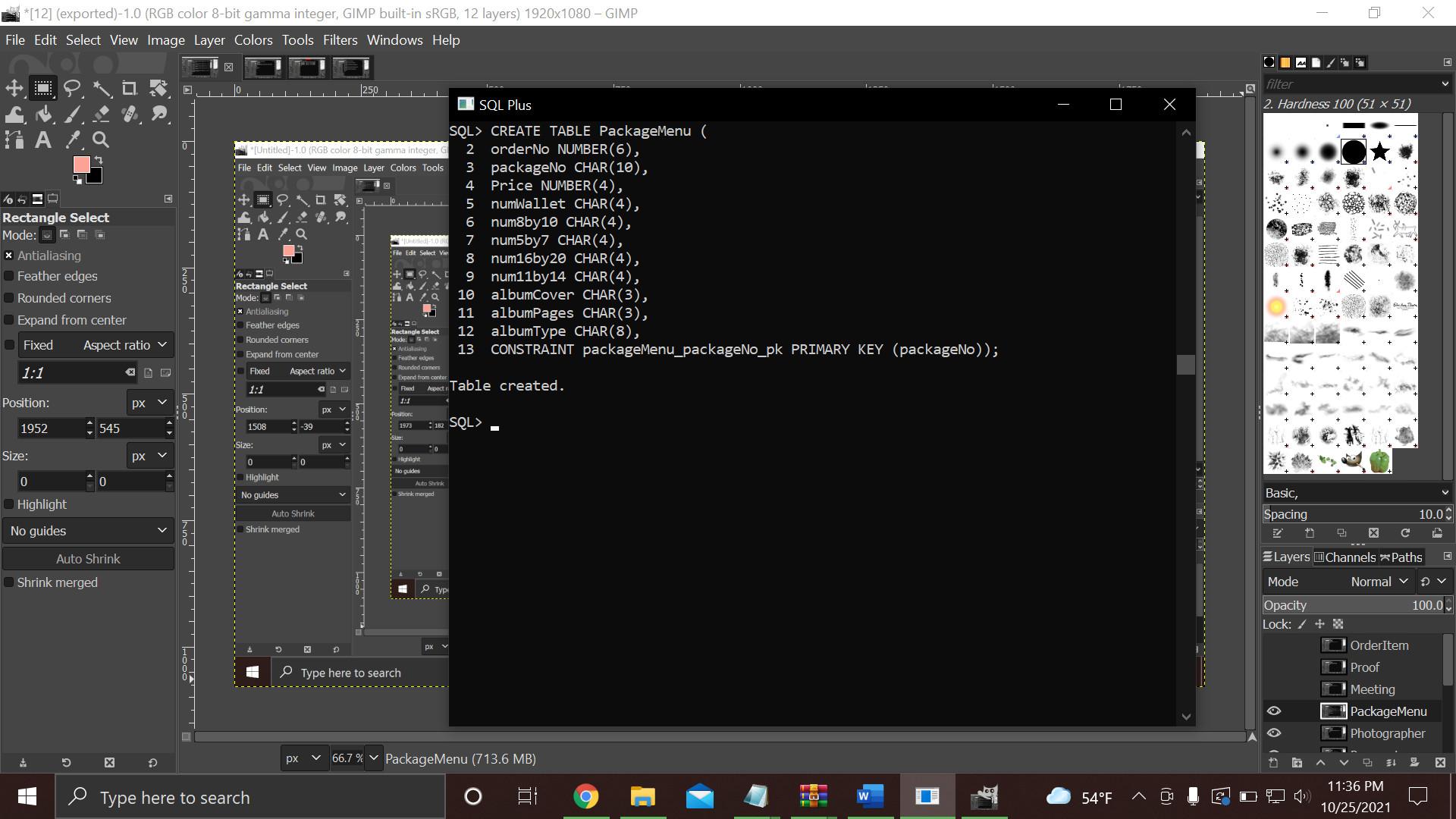
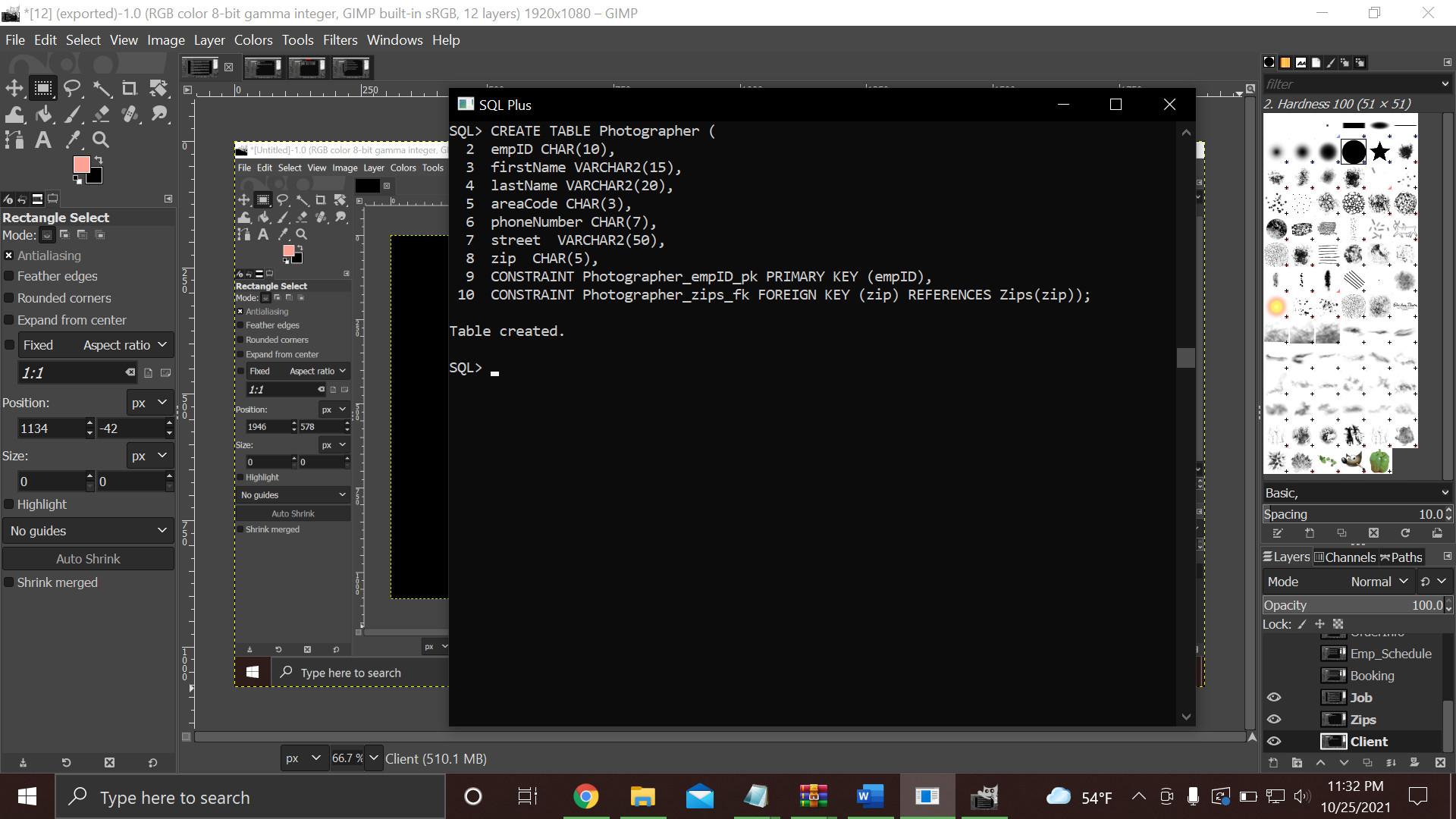
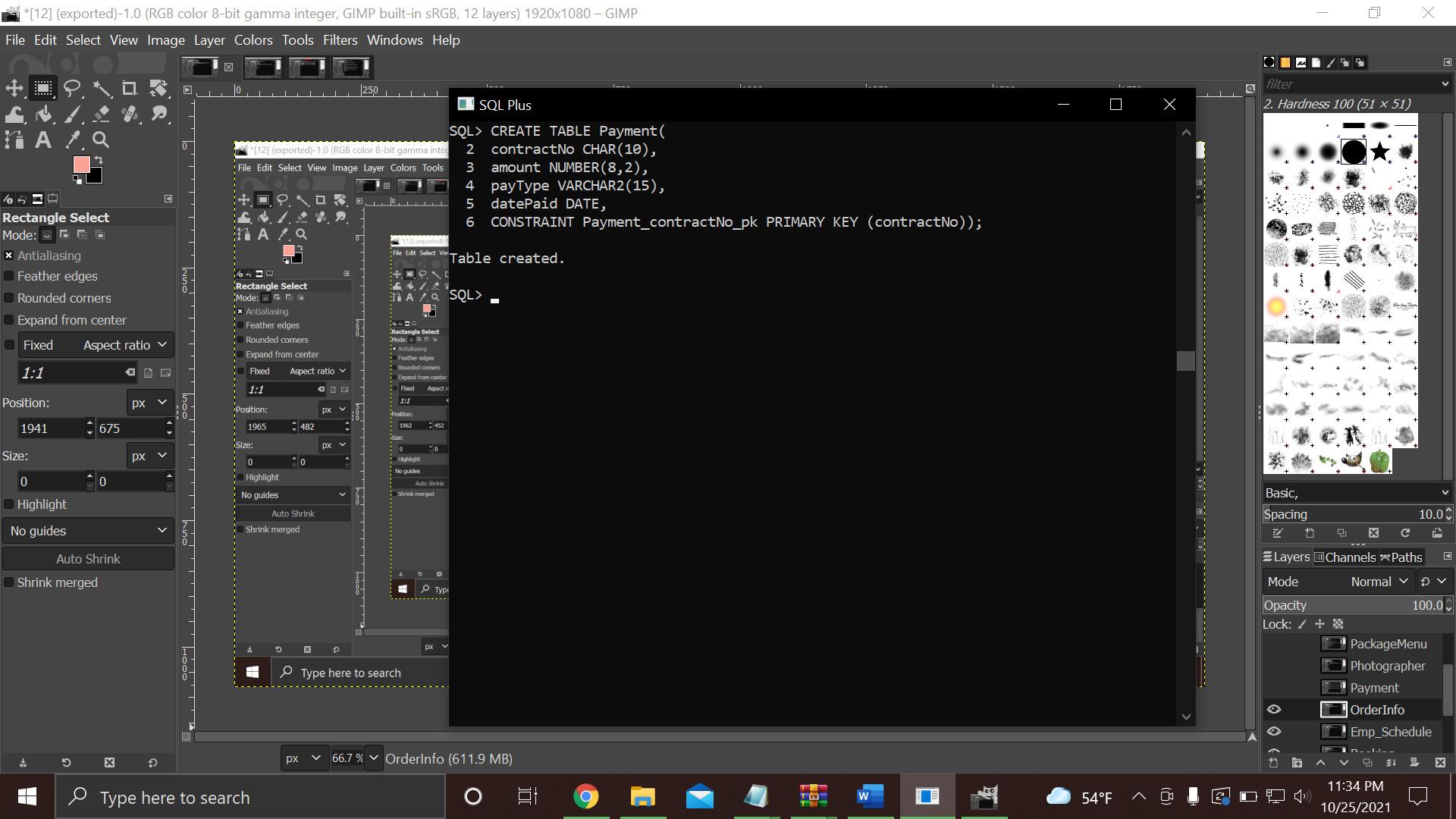
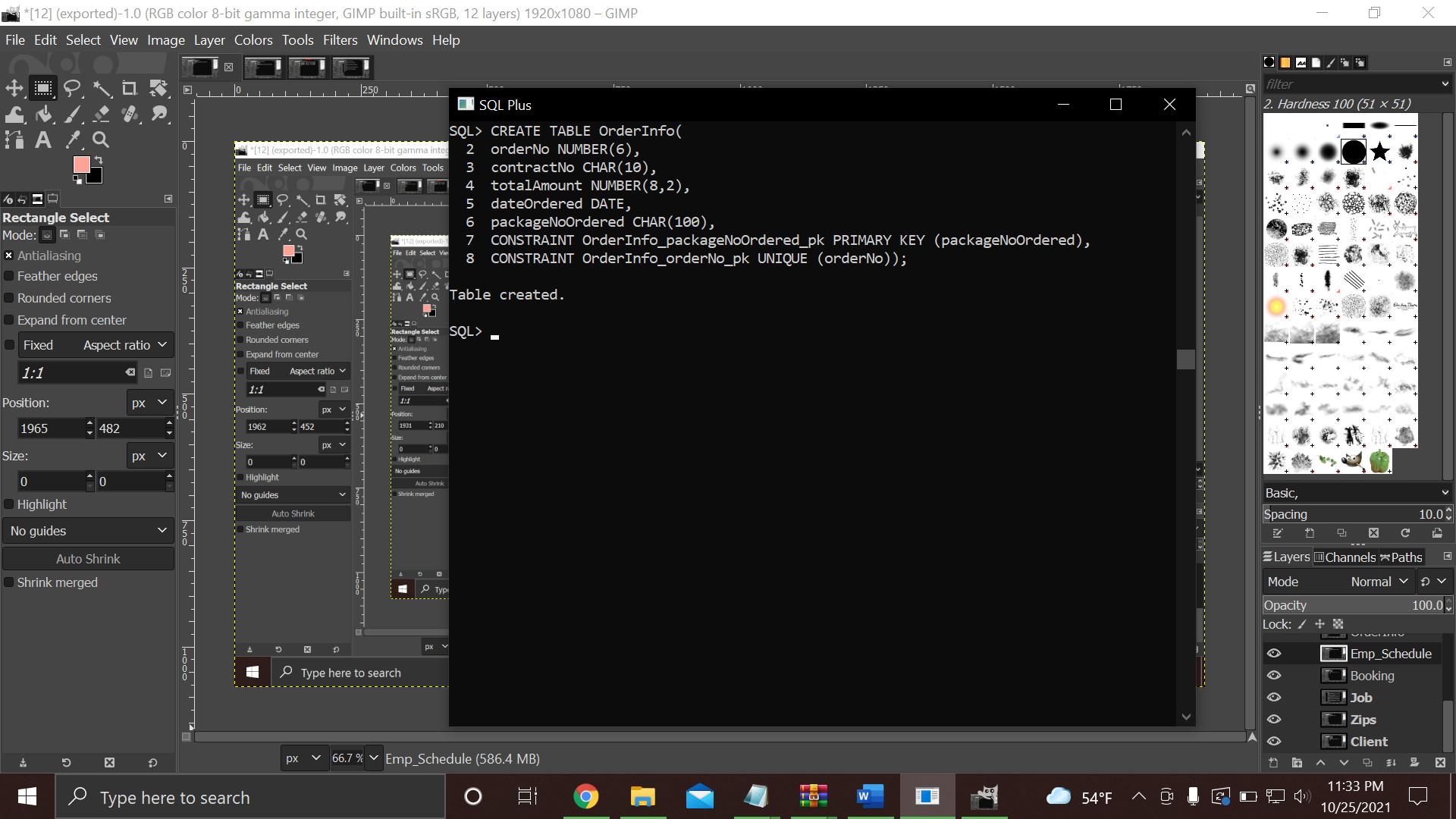
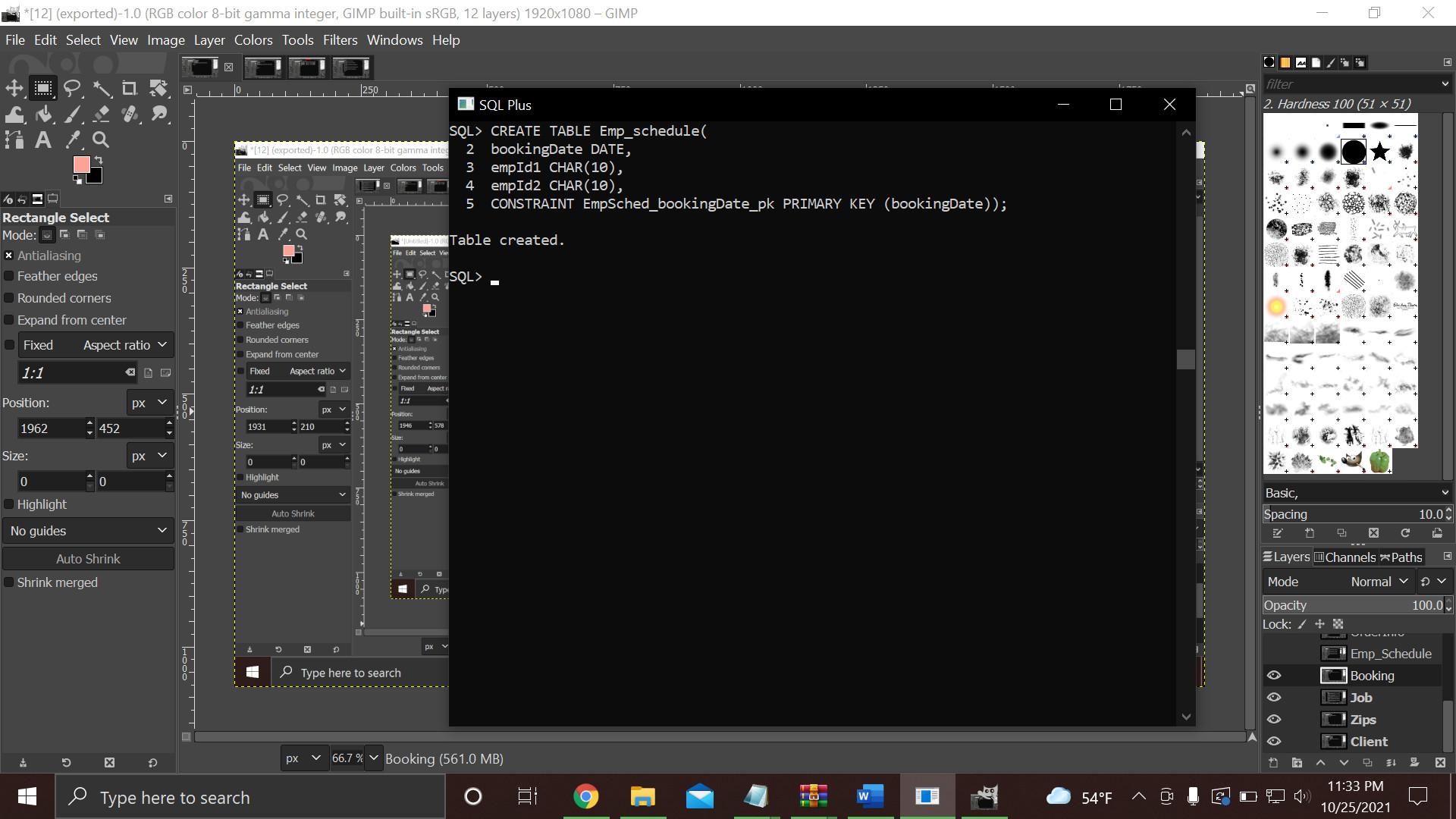
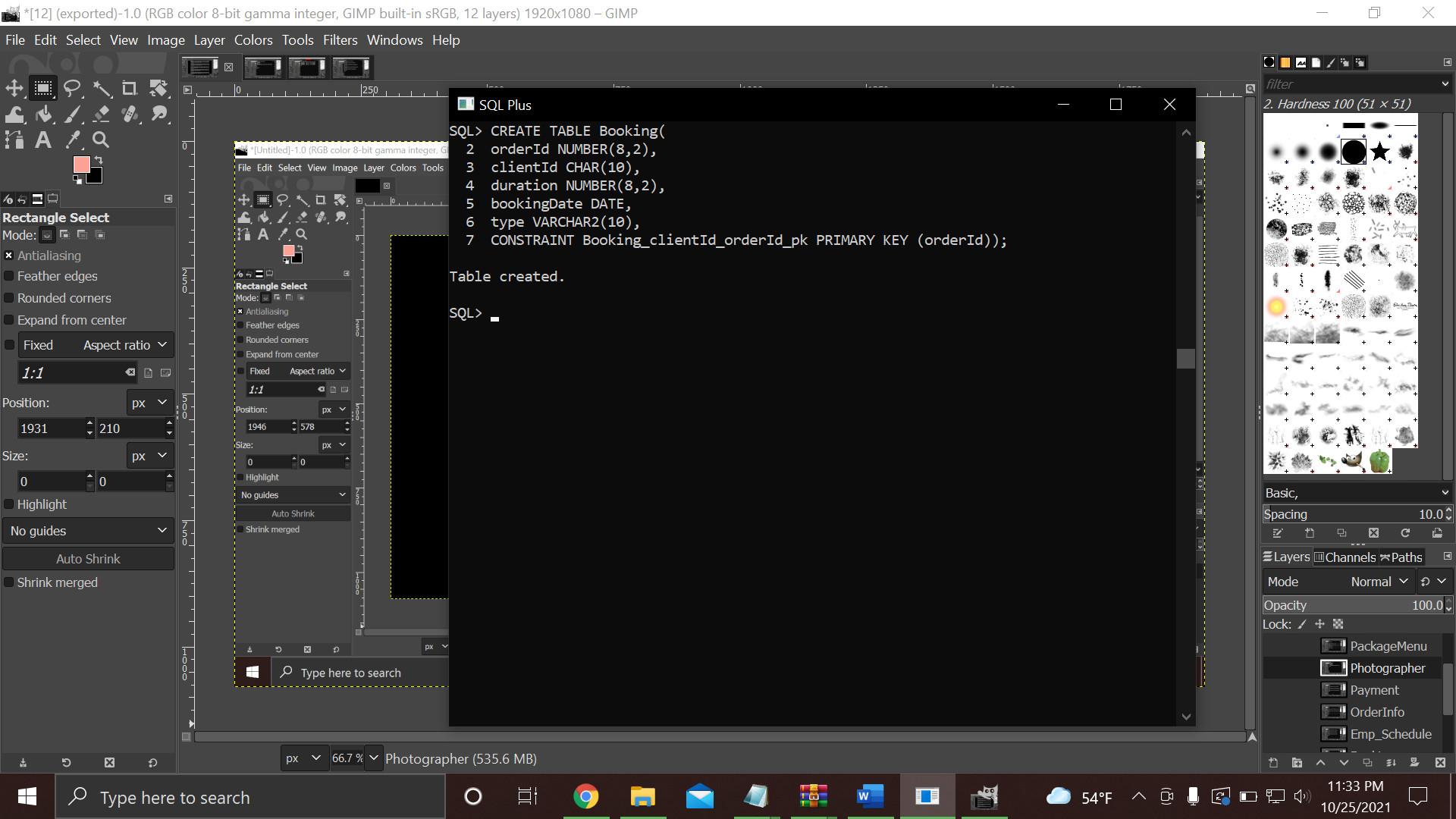
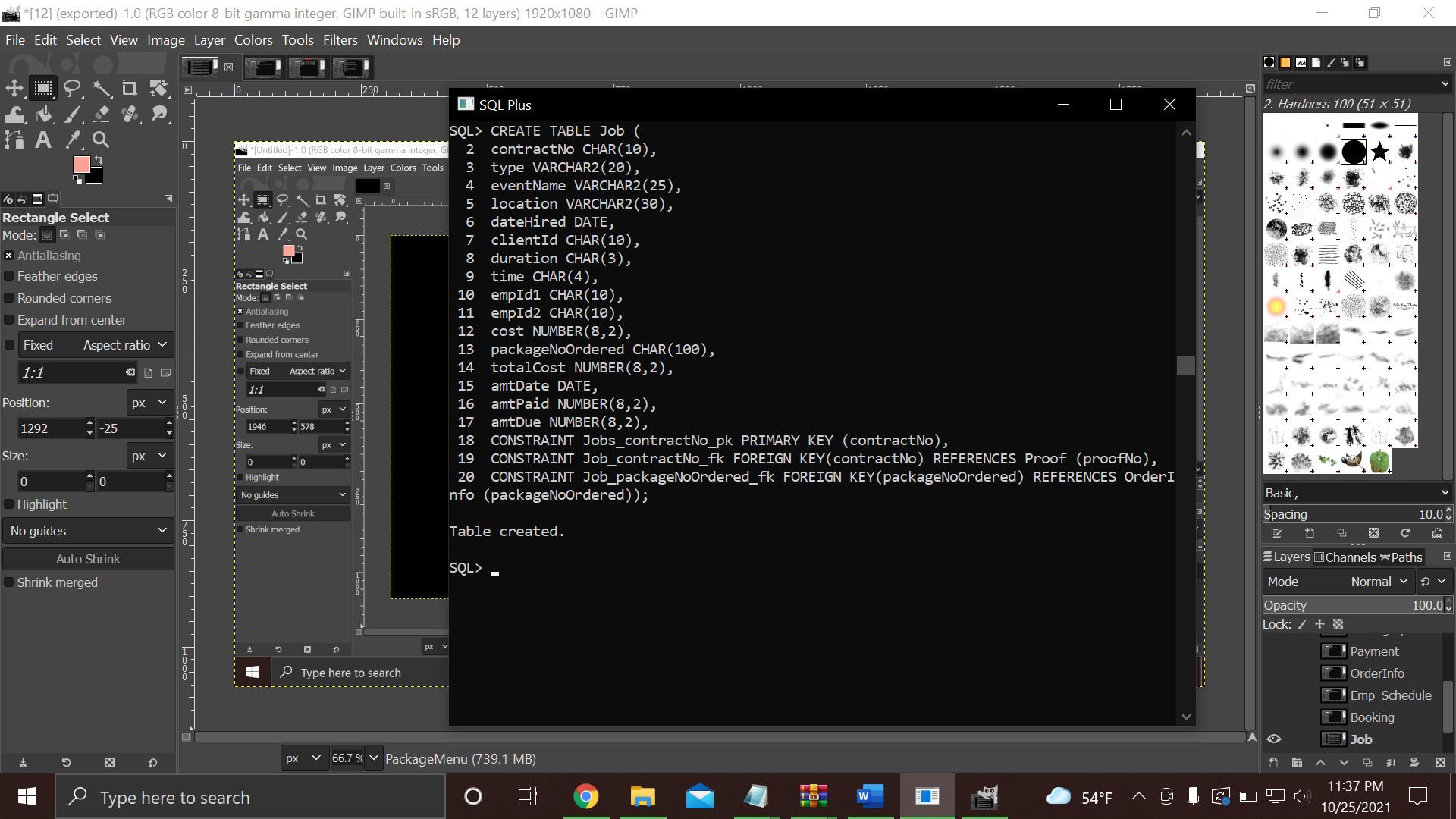
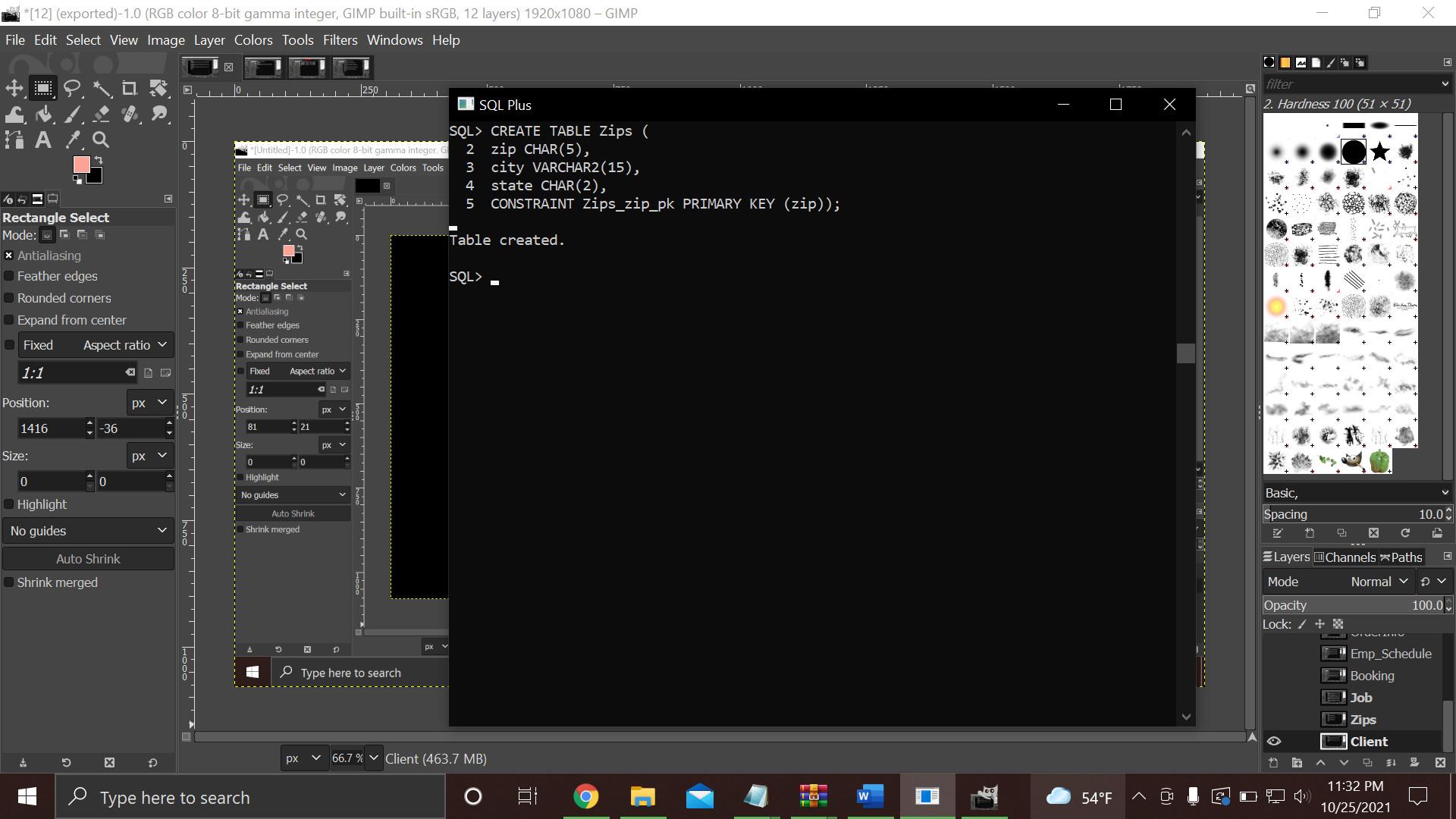
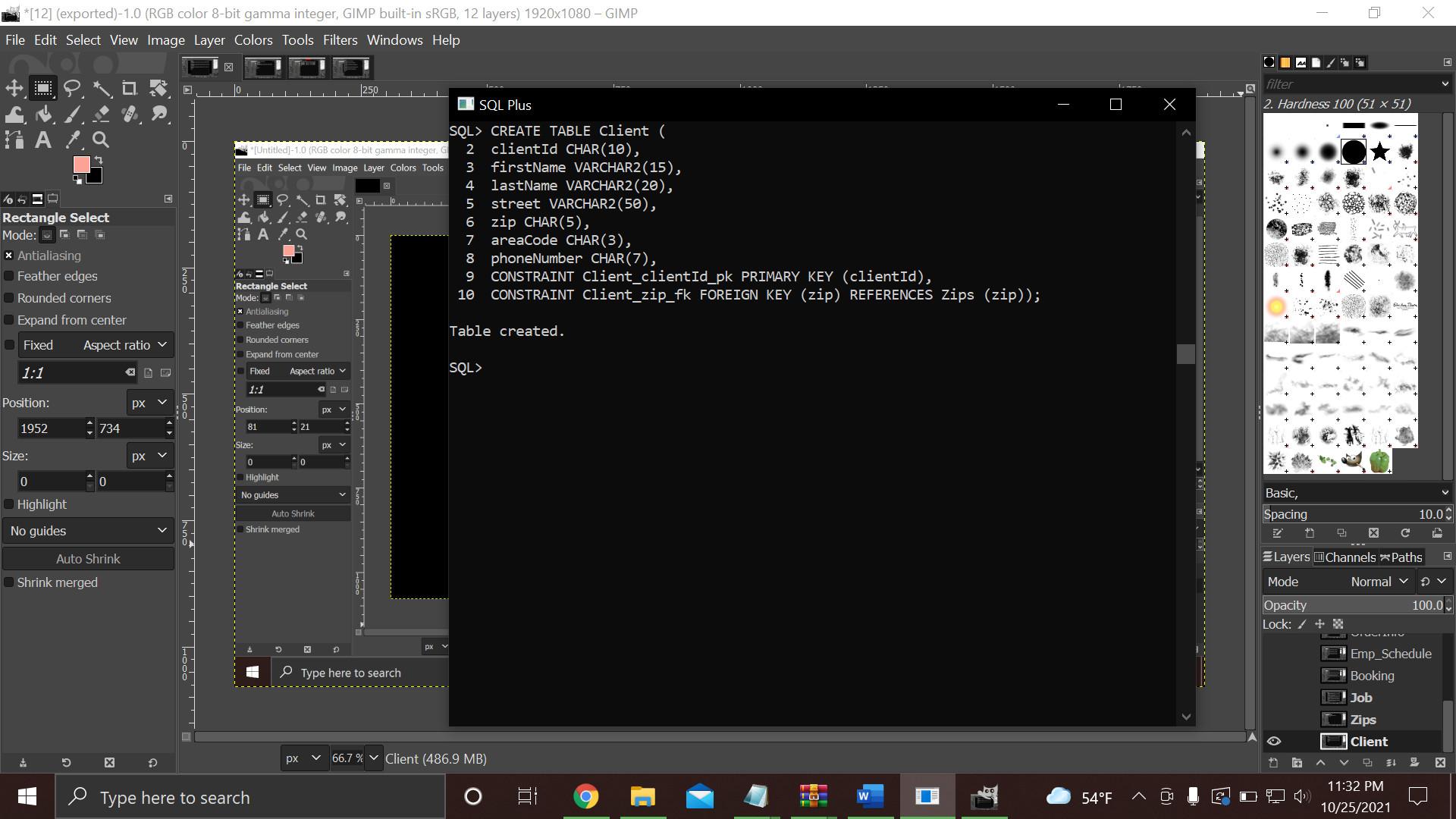
UPDATE PackageMenu

SET num11by14 = 2

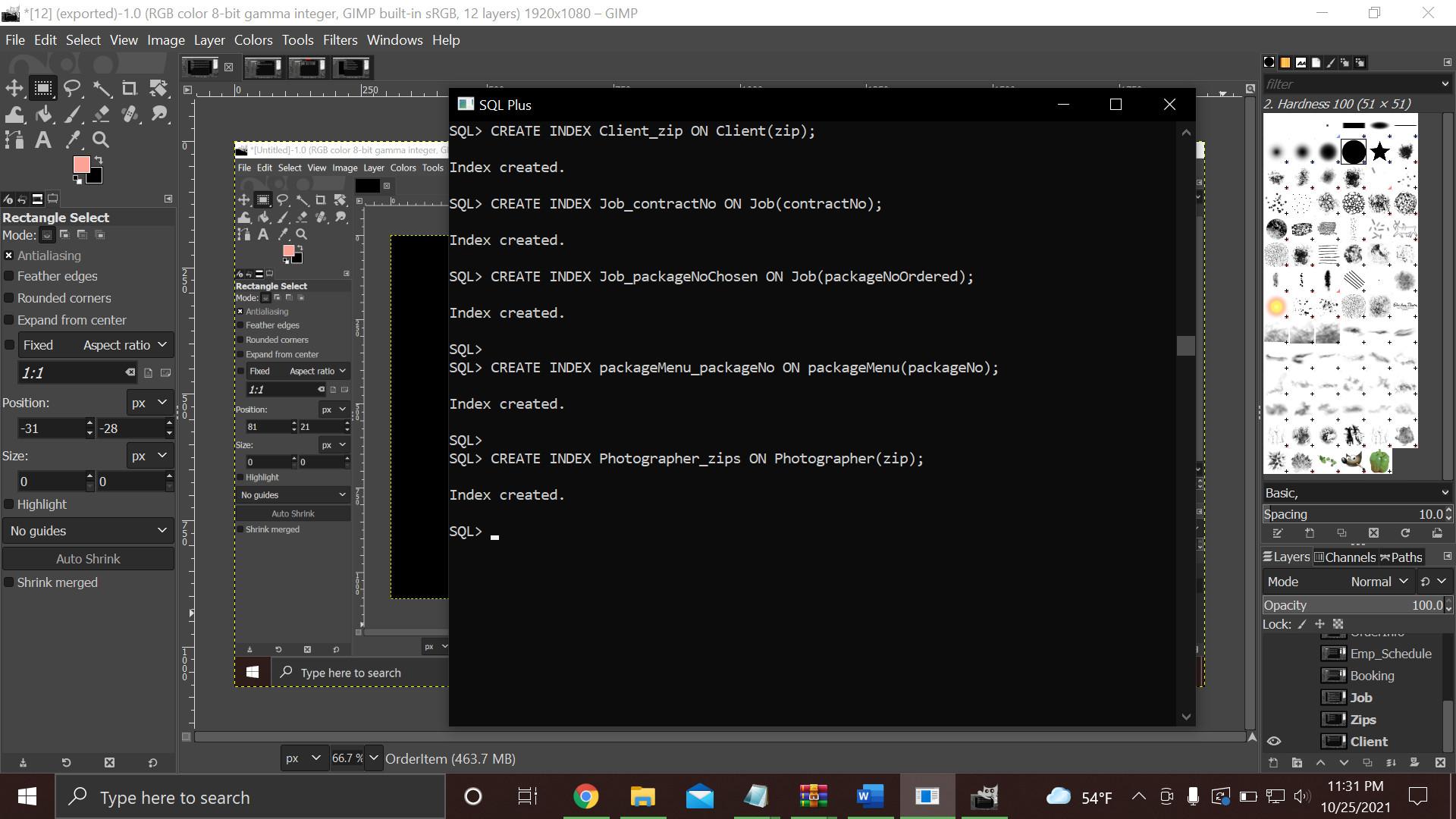
WHERE albumCover = 'NO';

Screenshots:

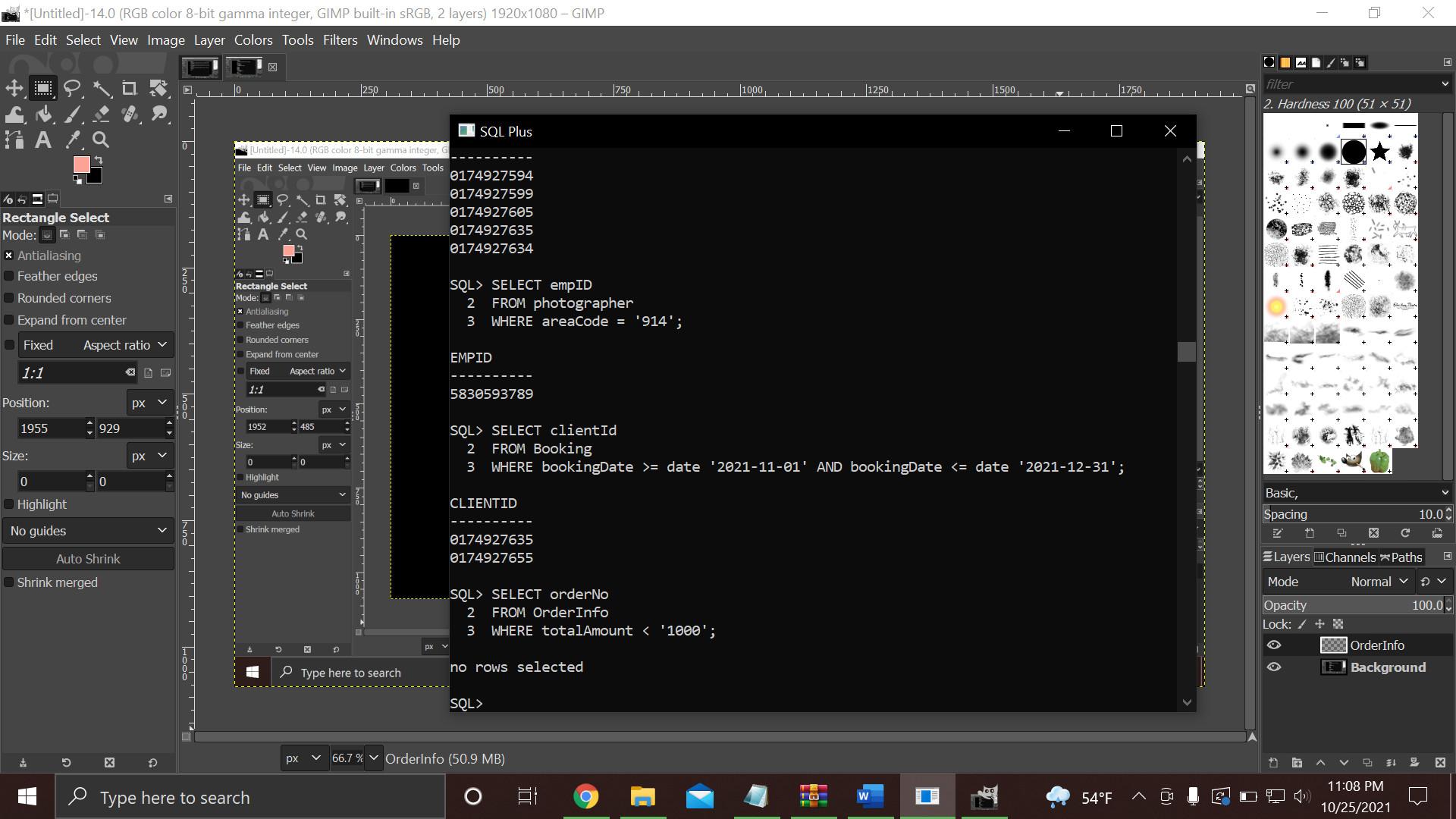
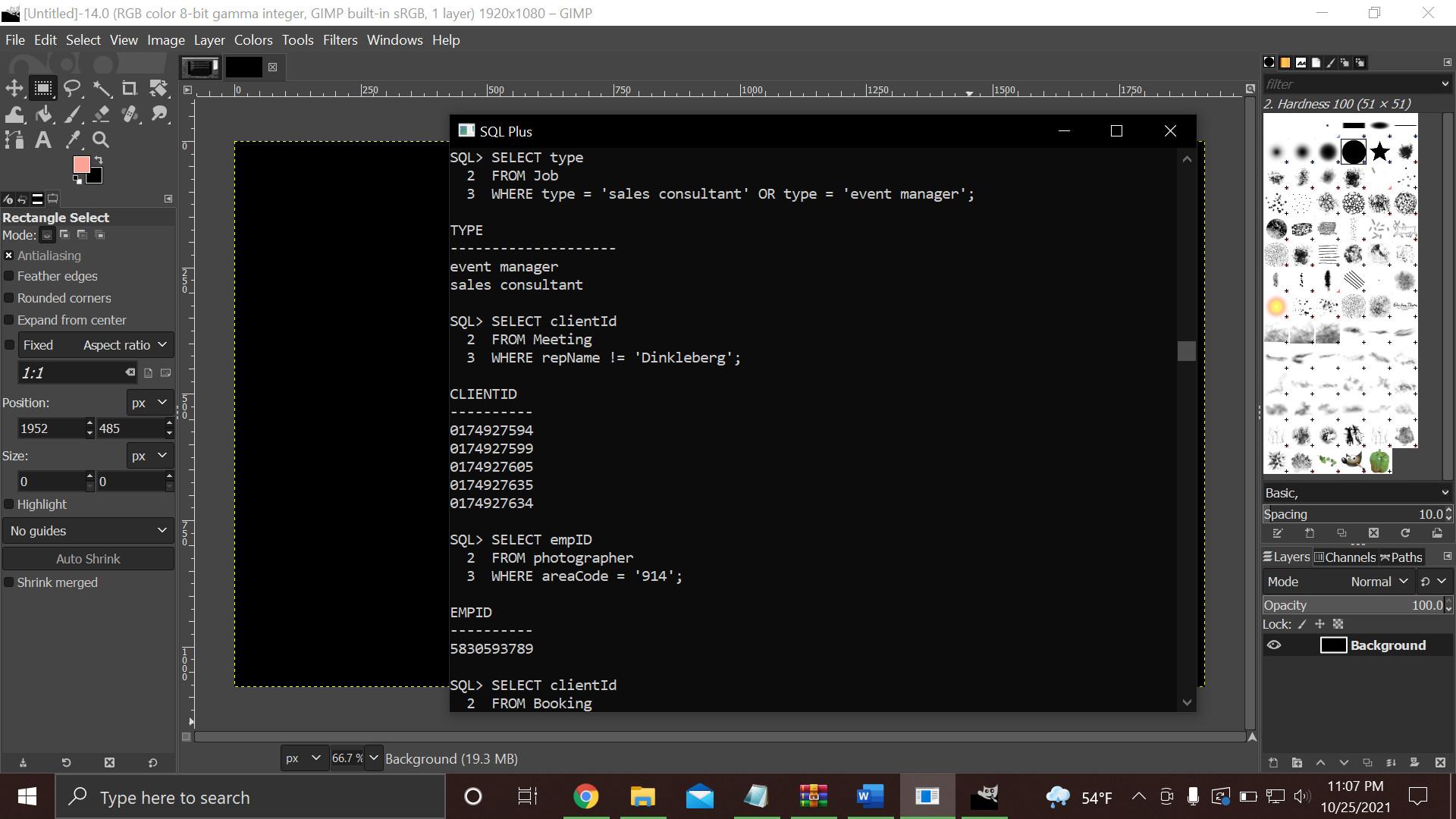
6.4:



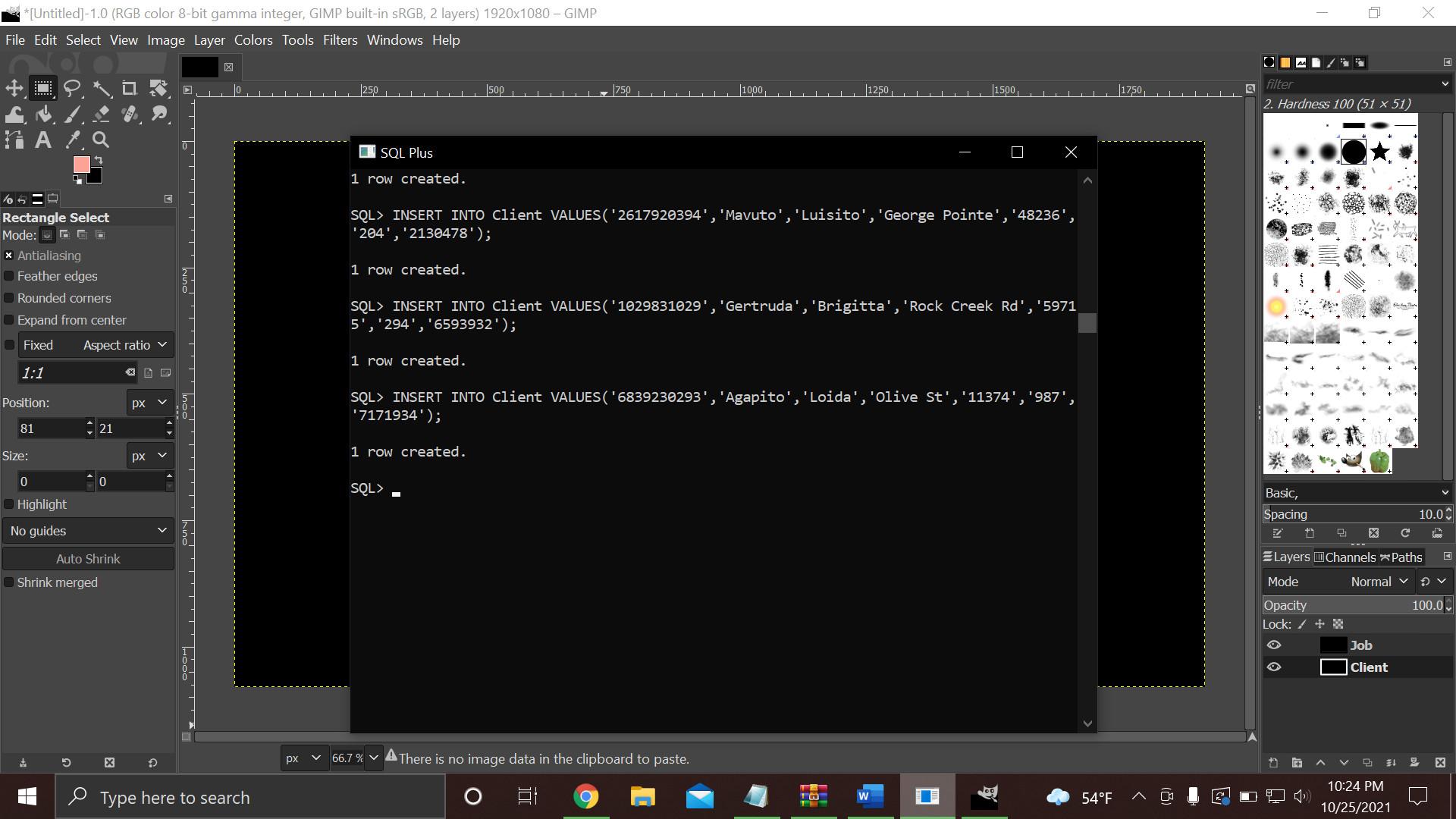
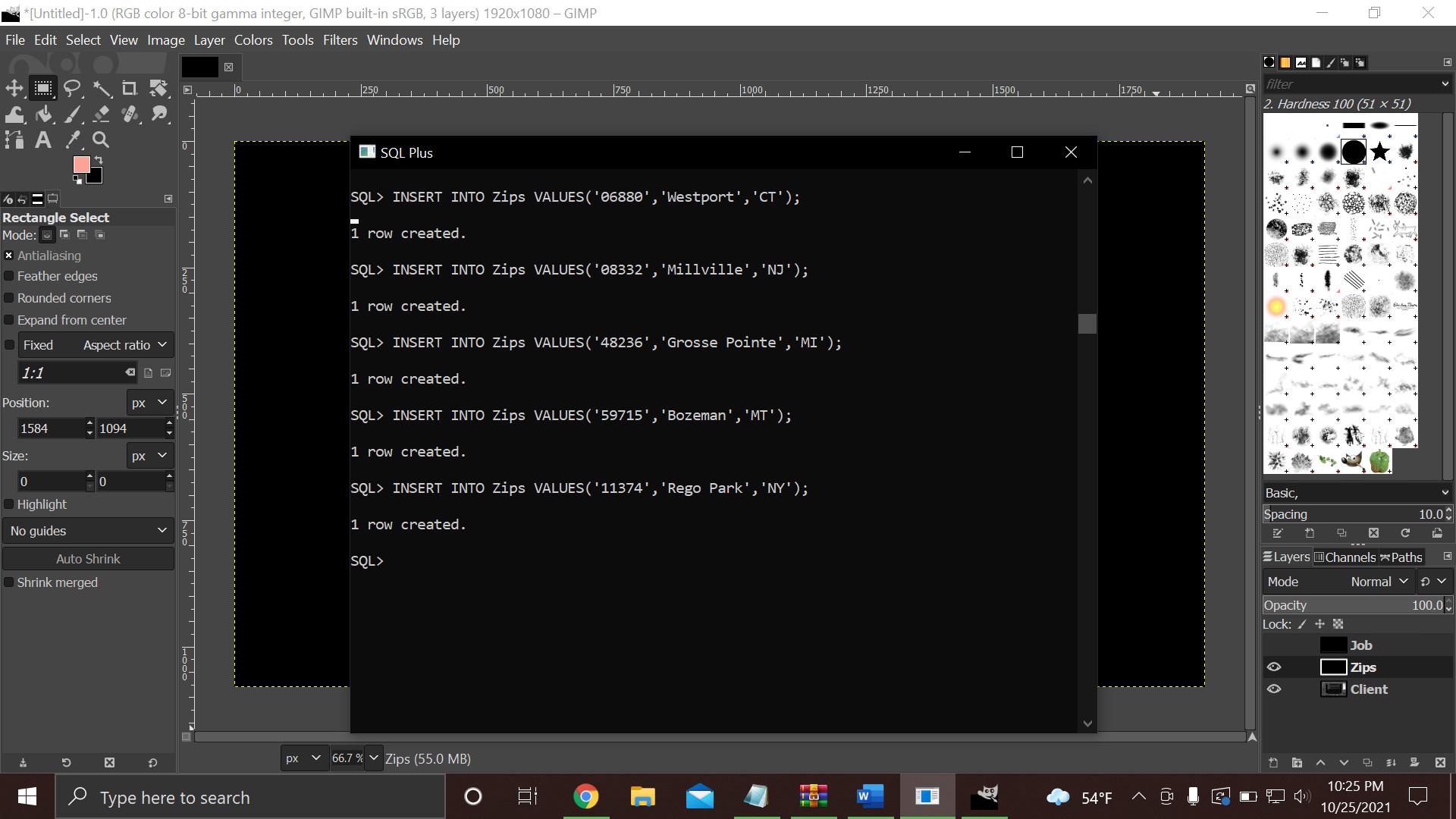
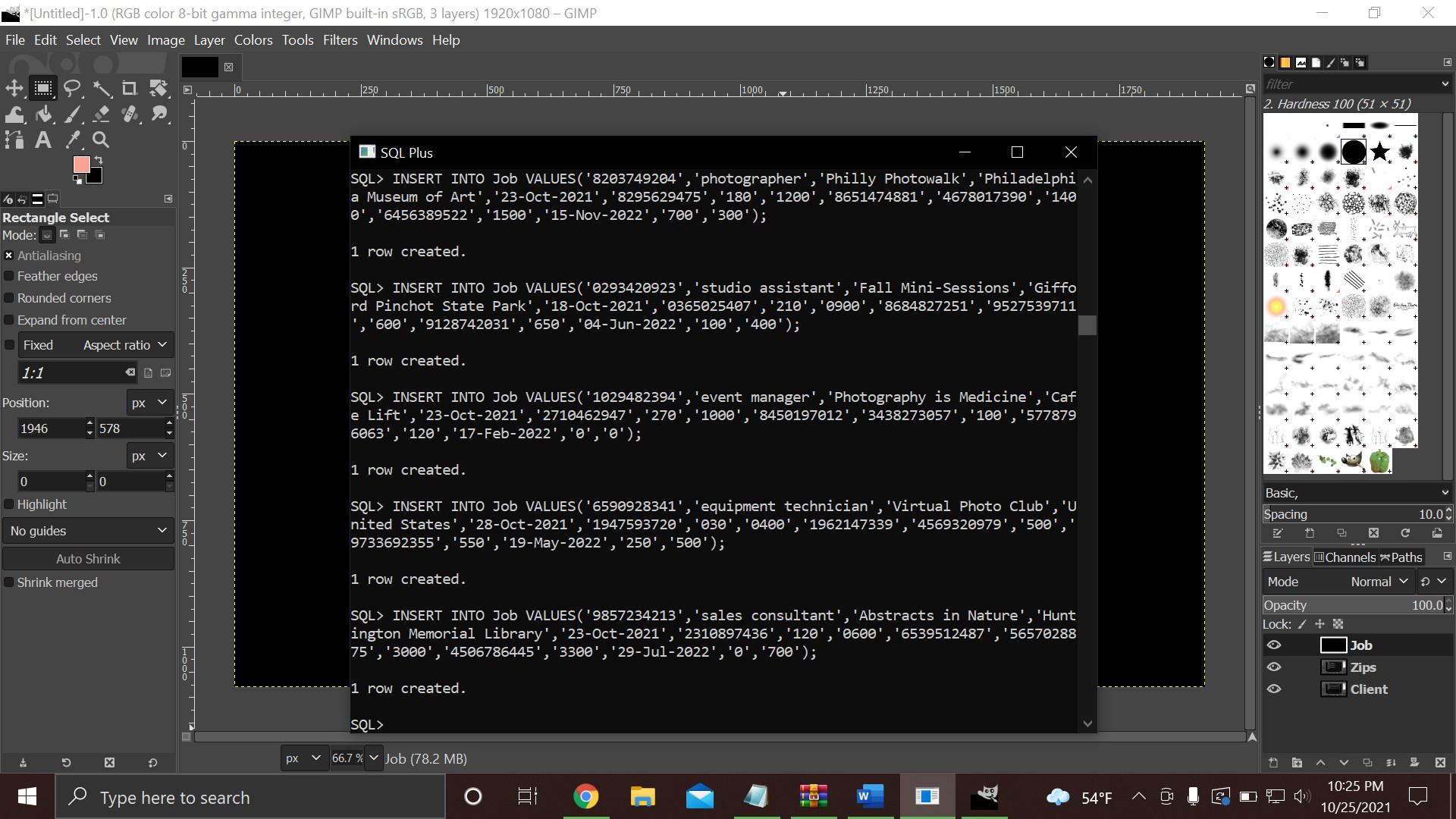
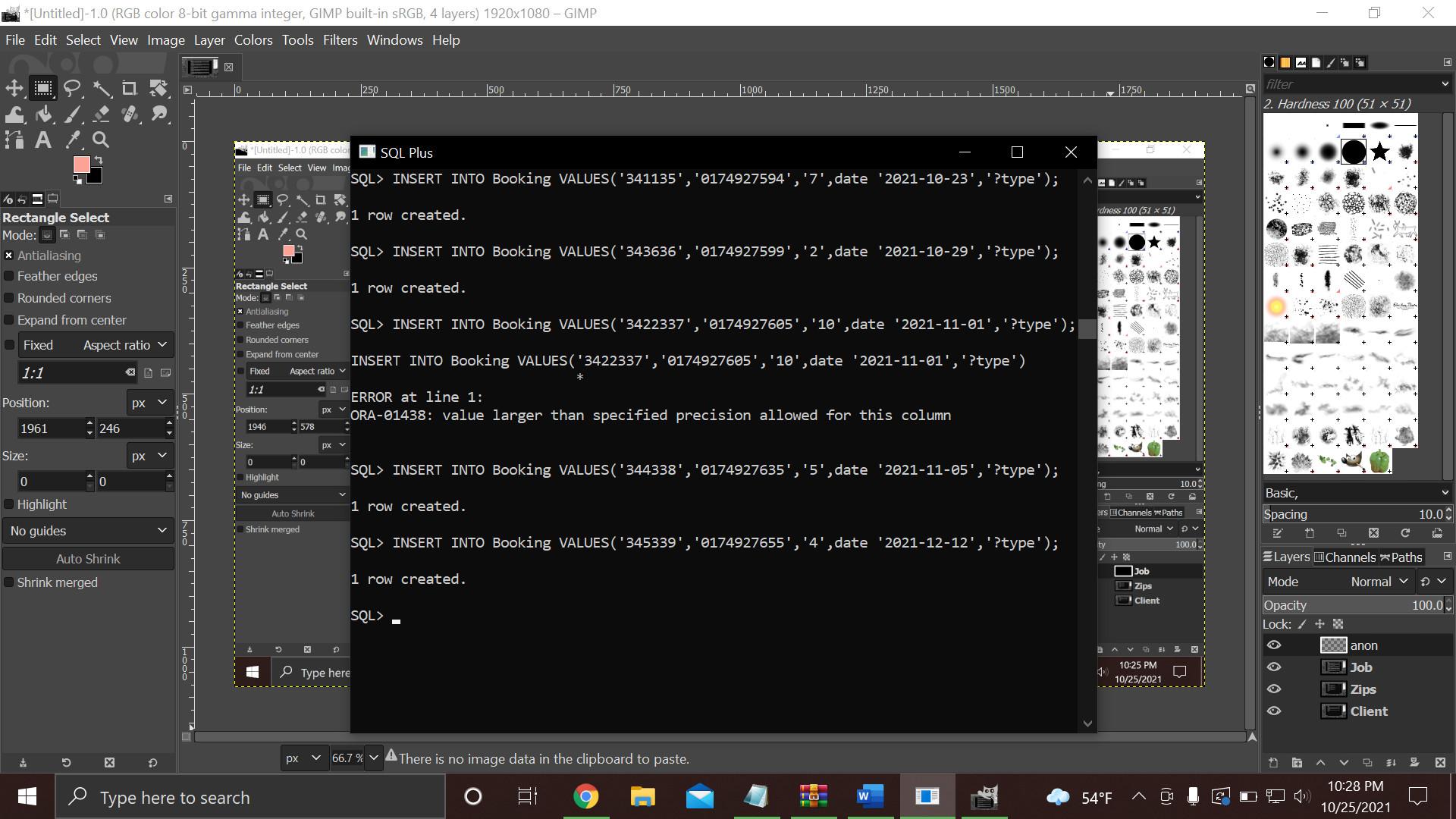
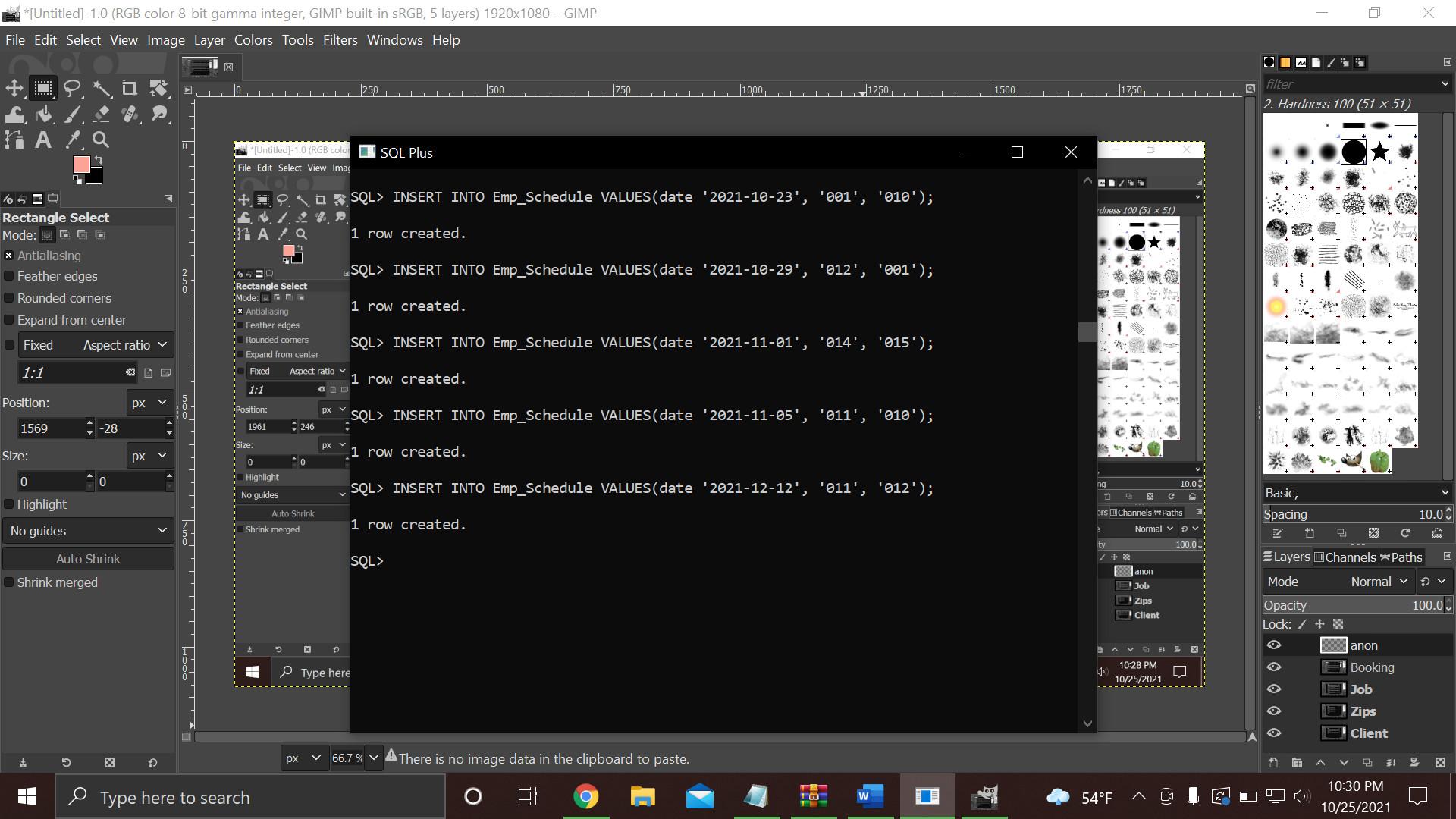
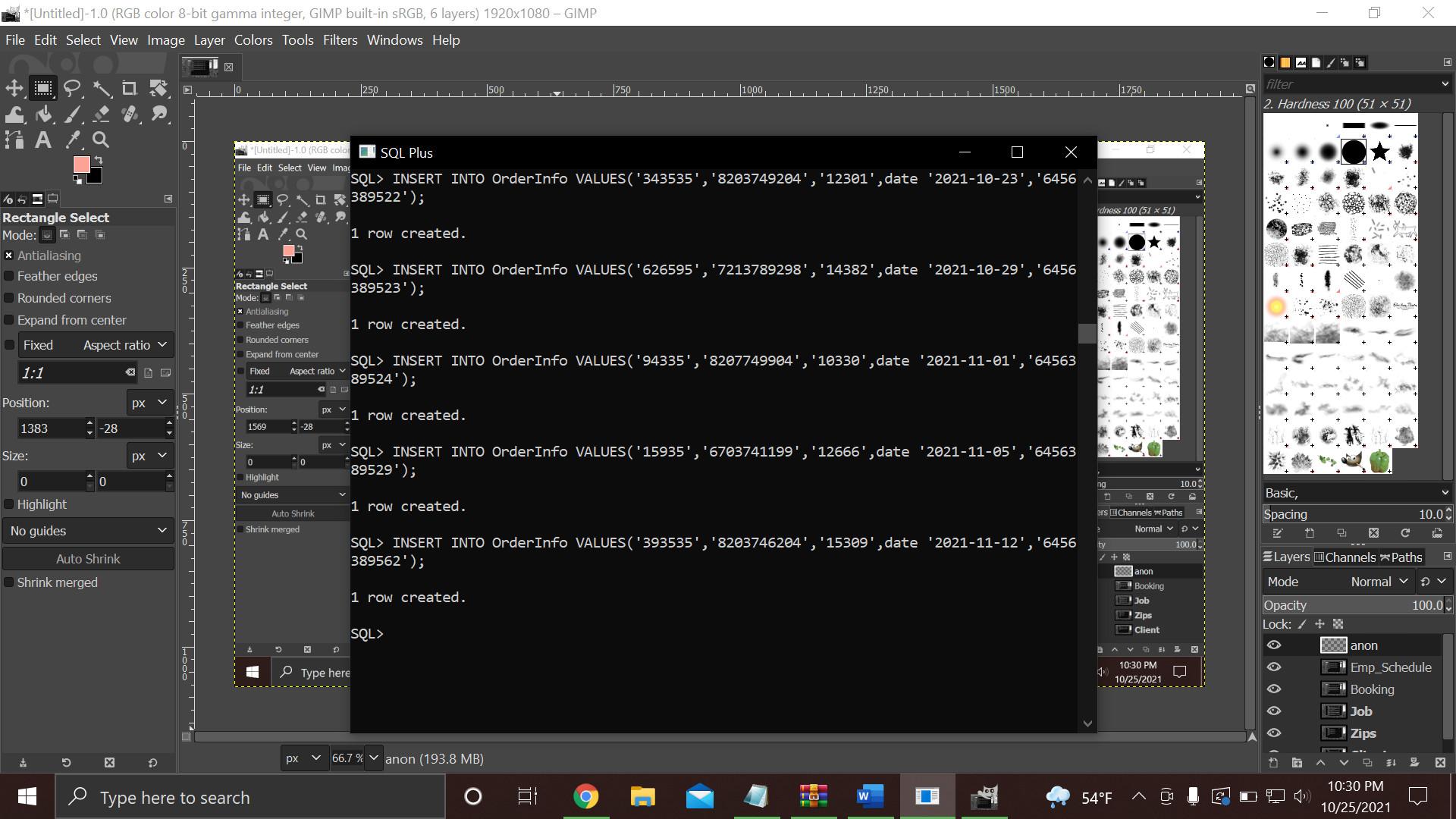
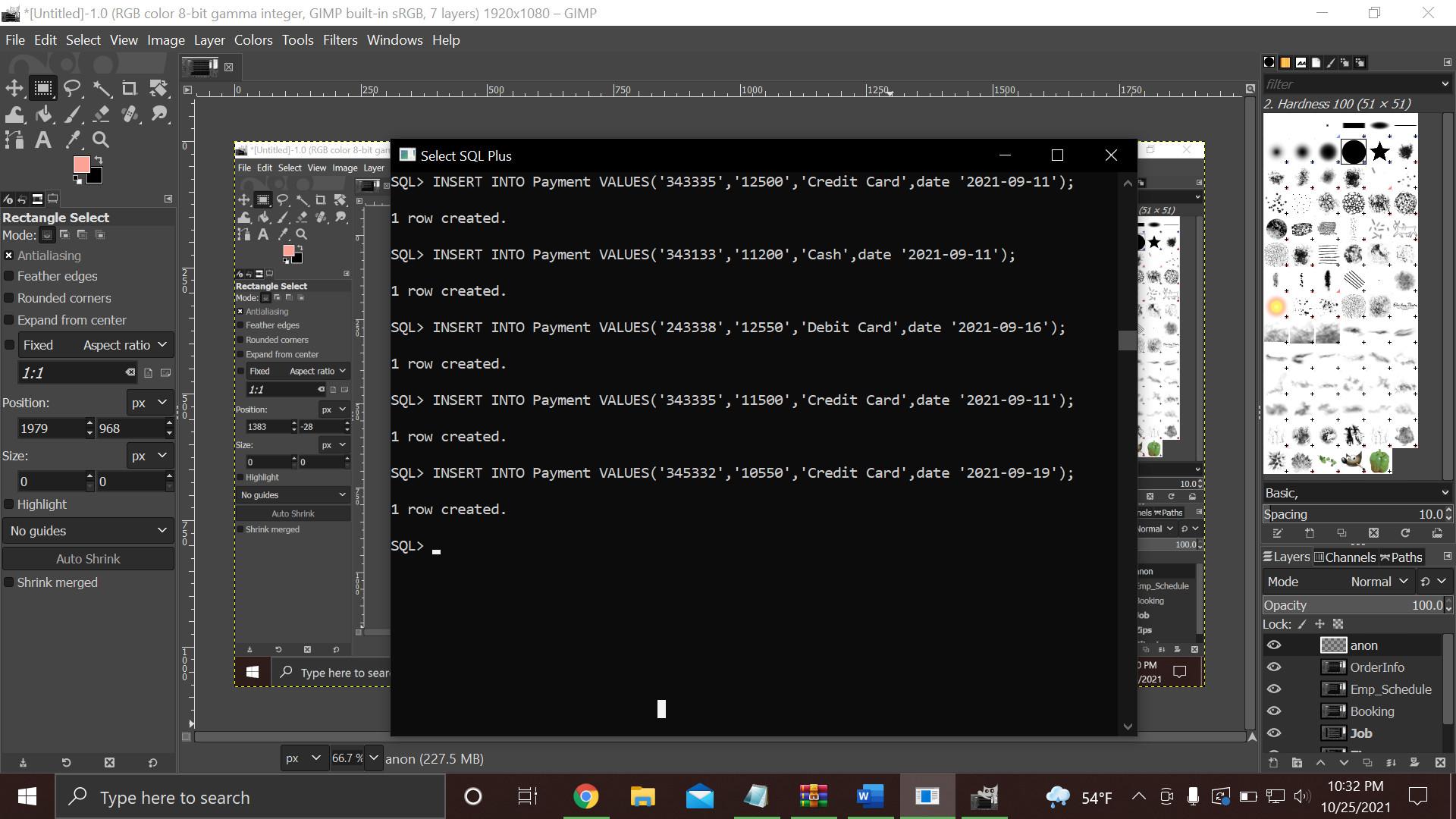
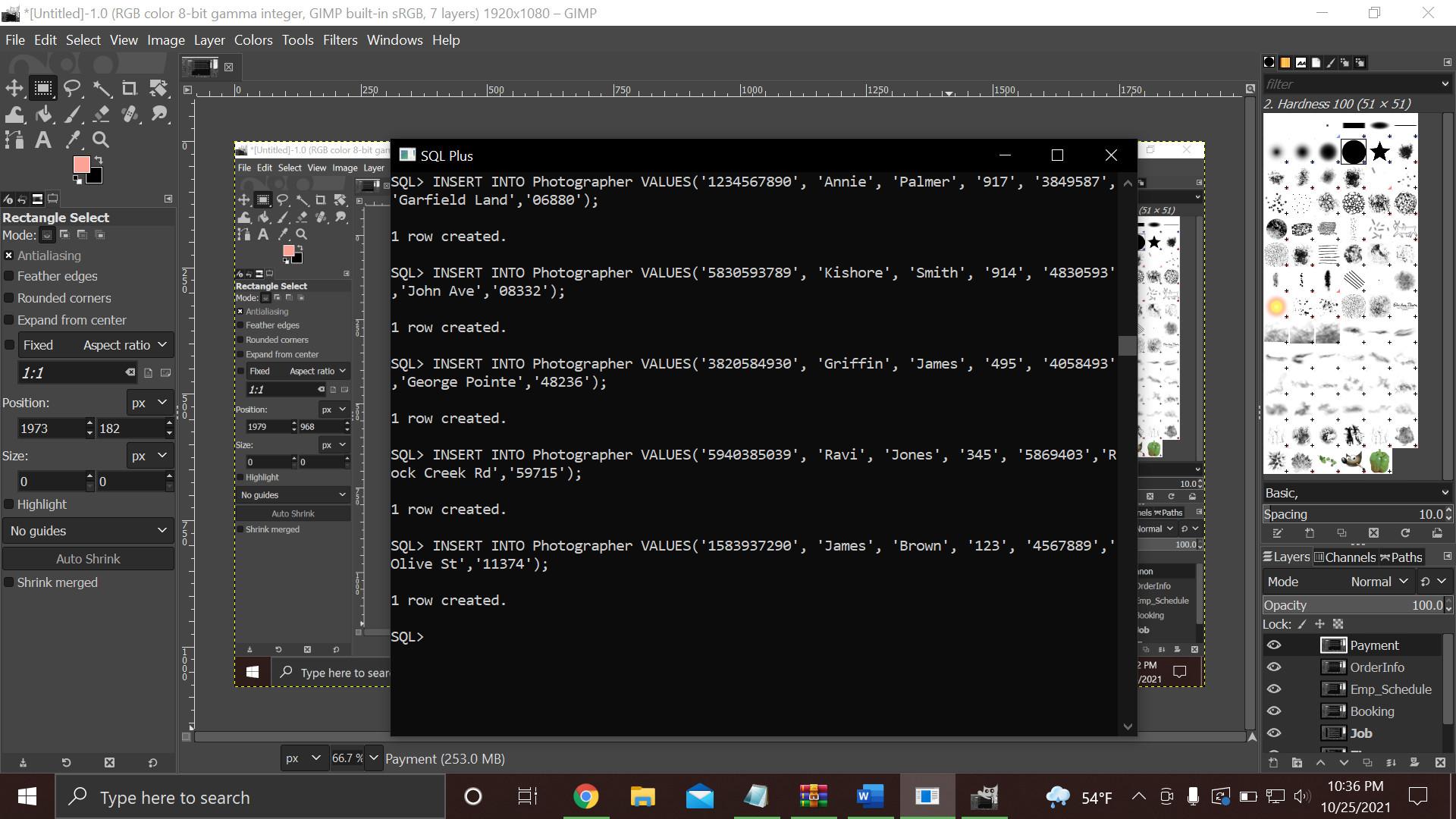
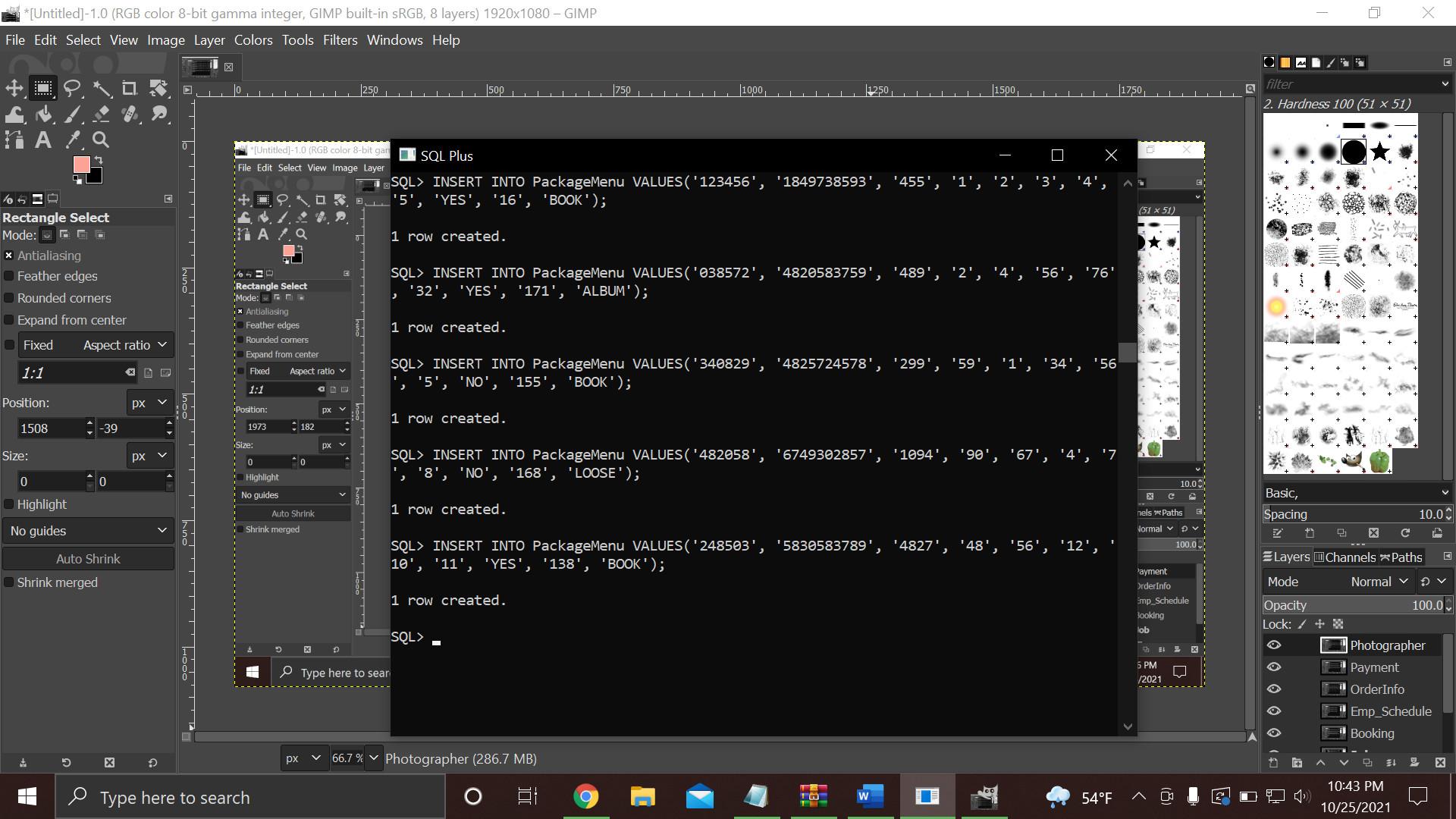
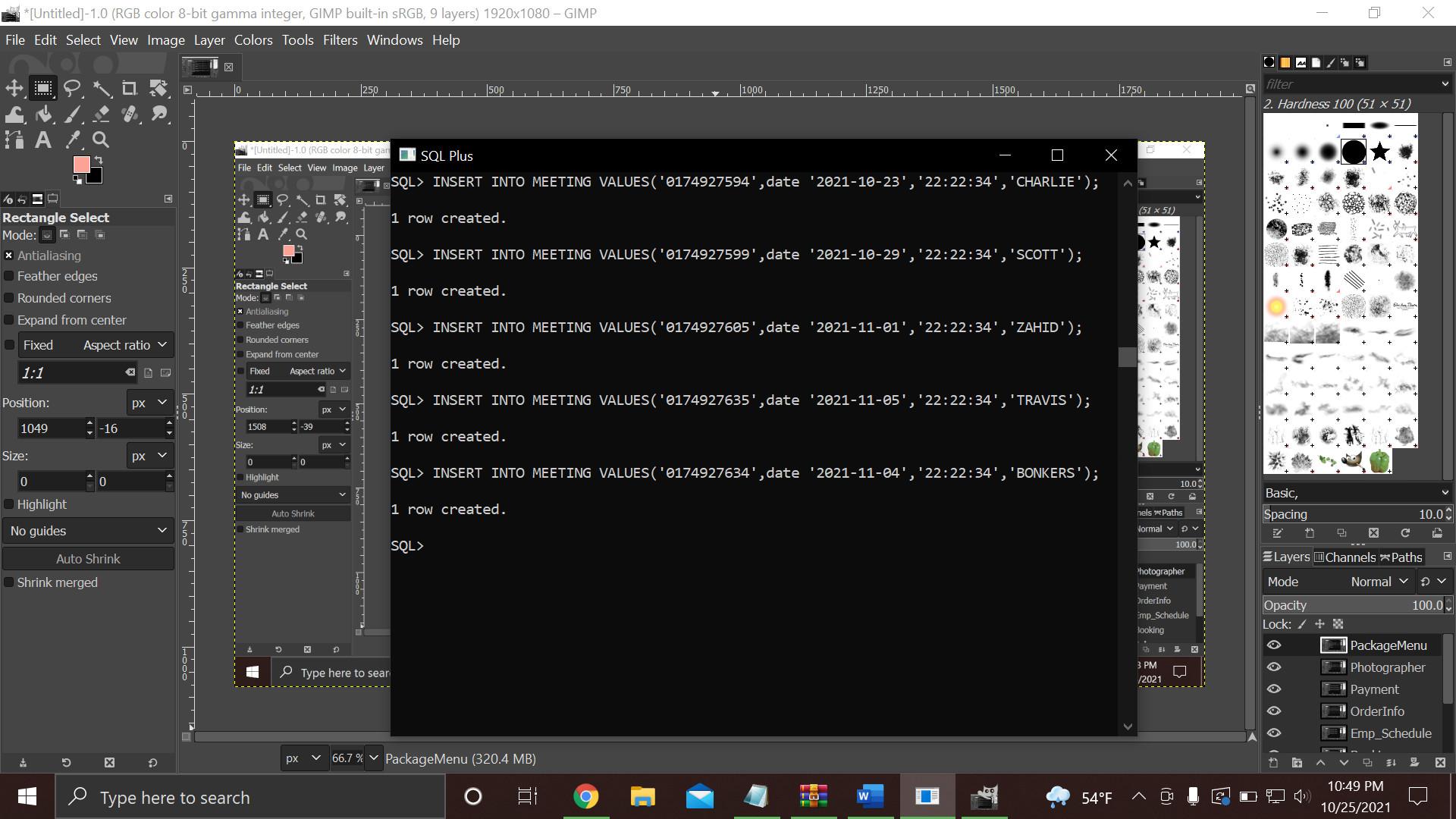
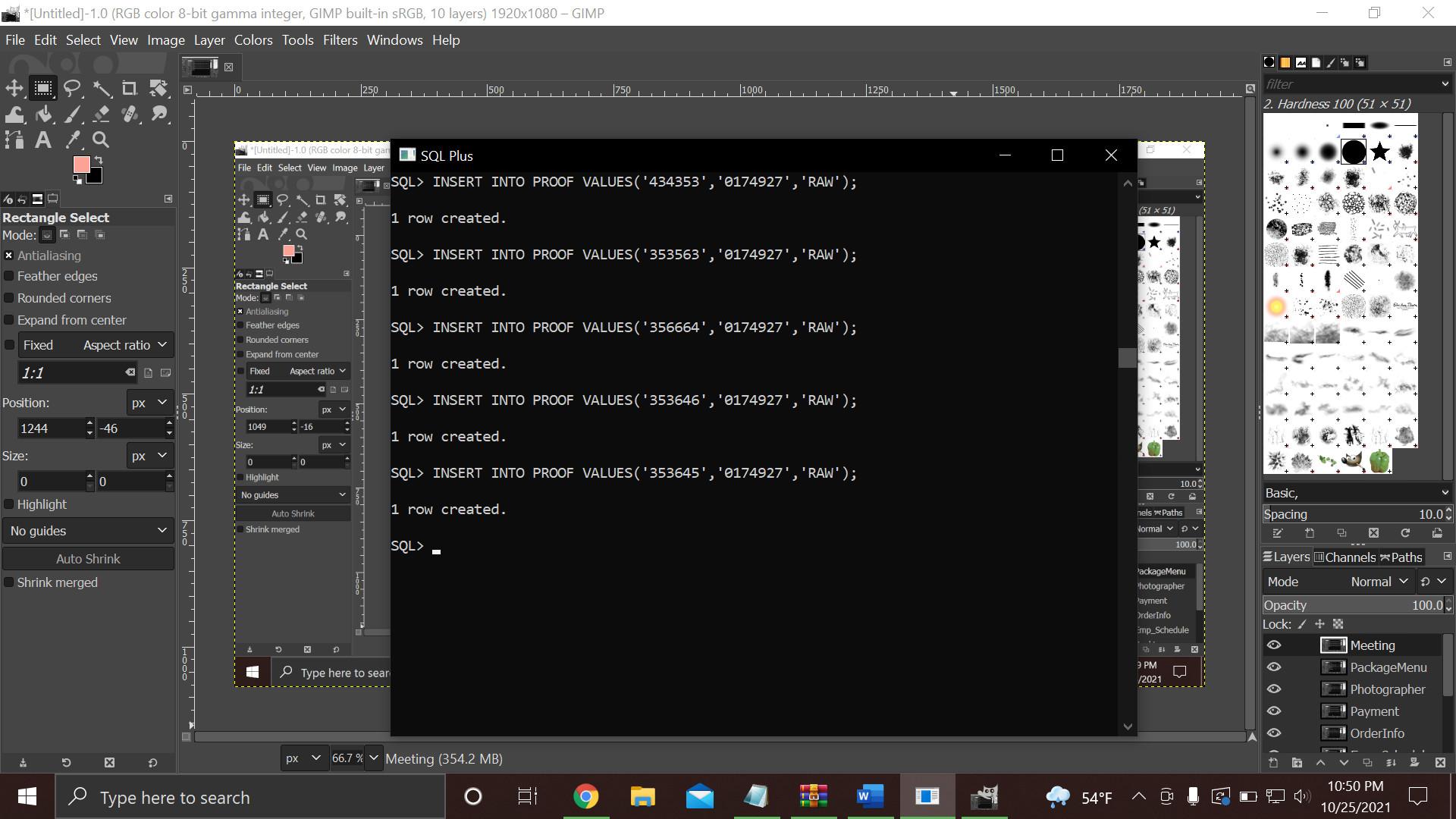
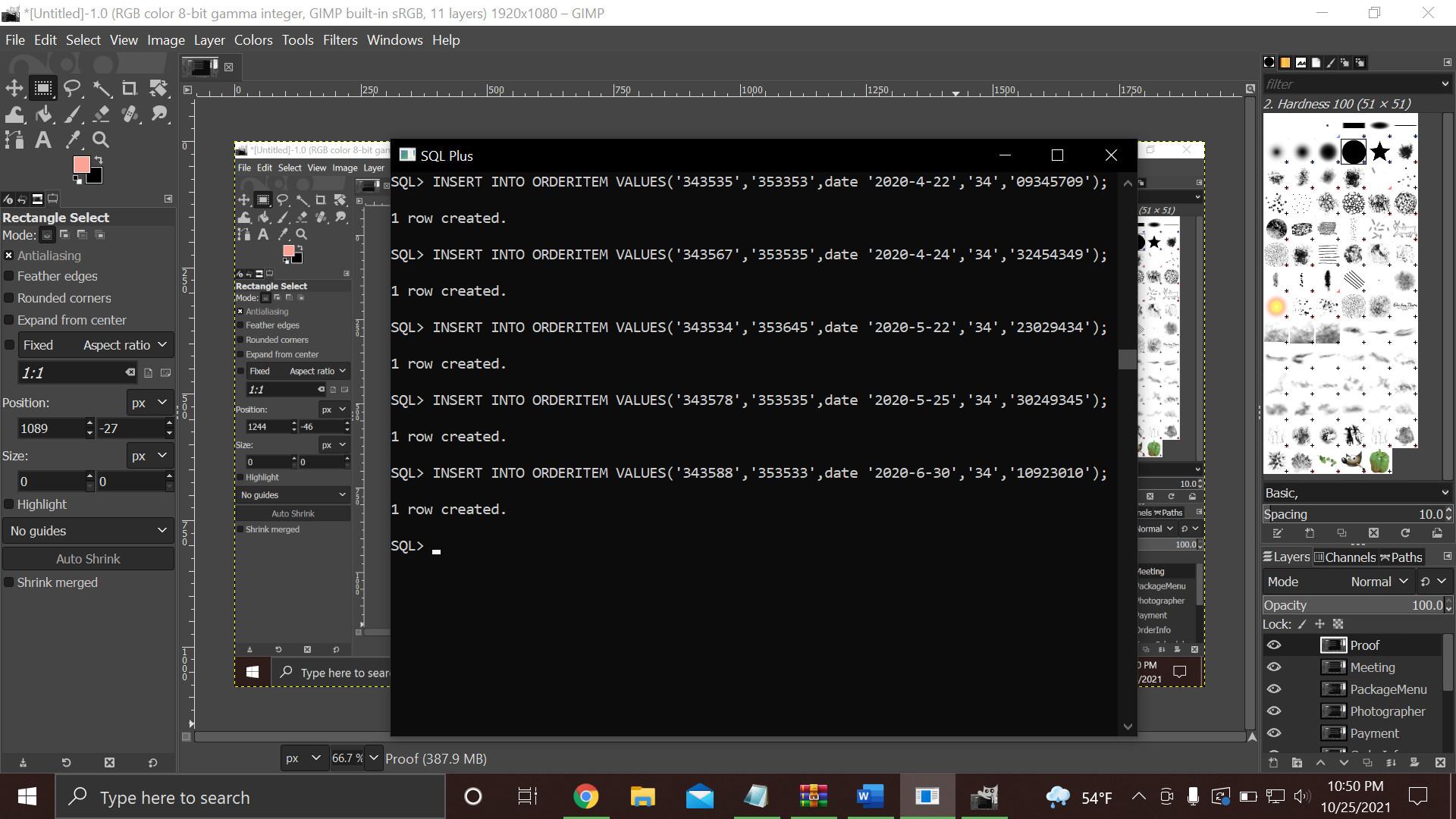
6.5



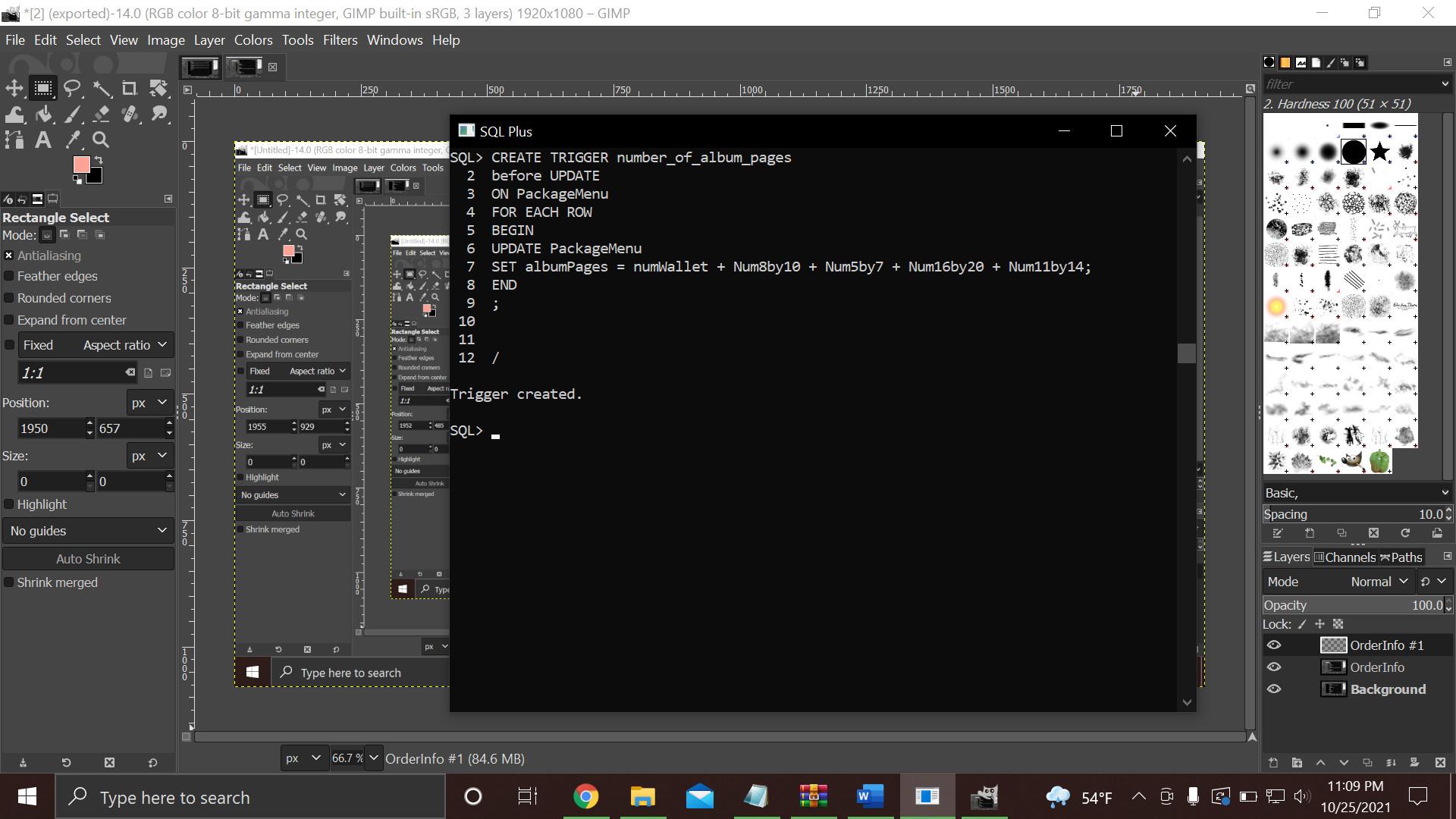
6.6



6.7



6.8



6.9

