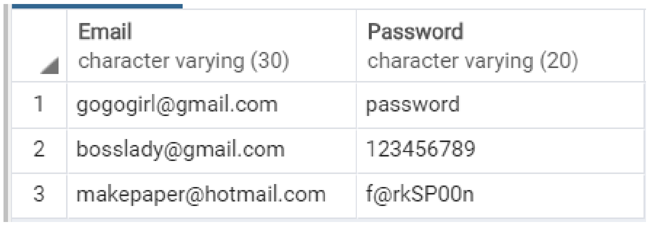
CSE 305 - Homework 3 Report

By Jason Dong, Agnieszka Gawrys, and Melanie Logan

ACCOUNT: An account has a email as its primary key. Email has a check requiring it contains a "@" character. There is also a check on password so that it is at least eight characters long. Account is the parent relation of User_Account and Employee_Account which are discussed further below.

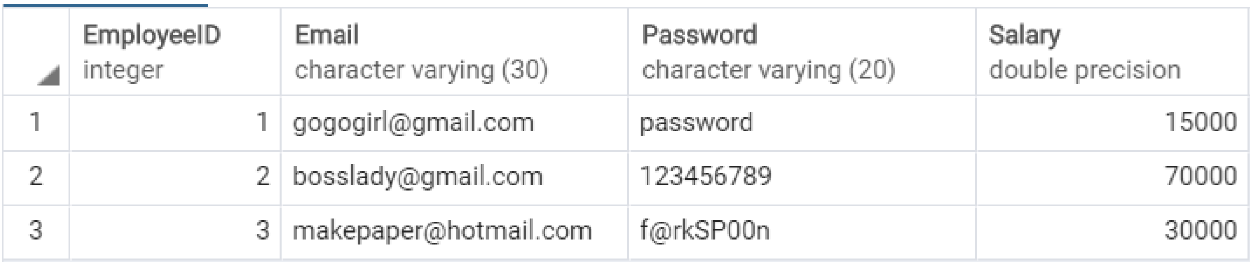
```
CREATE TABLE public."Account"
(
   "Email" character varying NOT NULL,
   "Password" character varying NOT NULL,
   CONSTRAINT "account_pkey" PRIMARY KEY ("Email"),
   CONSTRAINT password_check CHECK (length("Password"::text) >= 8),
   CONSTRAINT "email_check" CHECK ("Email"::text ~~ '%@%'::text)
)
```



	Email character varying (30)	Password character varying (20)
1	gogogirl@gmail.com	password
2	bosslady@gmail.com	123456789
3	makepaper@hotmail.com	f@rkSP00n

EMPLOYEE_ACCOUNT: Employee_Account is a Account so it inherits the Email and Password attribute and its constraints. It also has the attribute EmployeeID which is unique and used to identify each employee without revealing their email. The last attribute is salary, represented by a double and can be null in the case a employee makes wage.

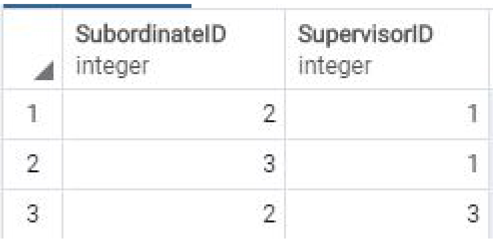
```
CREATE TABLE public."Employee_Account"
(
   "EmployeeID" integer NOT NULL,
   -- Inherited from table public."Account": "Email" character varying NOT NULL,
   -- Inherited from table public."Account": "Password" character varying NOT NULL,
   "Salary" double precision,
   CONSTRAINT "Employee_Account_pkey" PRIMARY KEY ("Email"),
   CONSTRAINT employeeid_unique UNIQUE ("EmployeeID")
,
   CONSTRAINT password_check CHECK (length("Password"::text) >= 8),
   CONSTRAINT email_check CHECK ("Email"::text ~~ '%@%'::text)
)
   INHERITS (public."Account")
```



	EmployeeID integer	Email character varying (30)	Password character varying (20)	Salary double precision
1	1	gogogirl@gmail.com	password	15000
2	2	bosslady@gmail.com	123456789	70000
3	3	makepaper@hotmail.com	f@rkSP00n	30000

REPORTSTO: This table has two attributes: SubordinateID and SupervisorID, both of which form the primary key and references an Employee_Account's EmployeeID. Both form the primary key because of the many-to-many relationship between subordinate and supervisor.

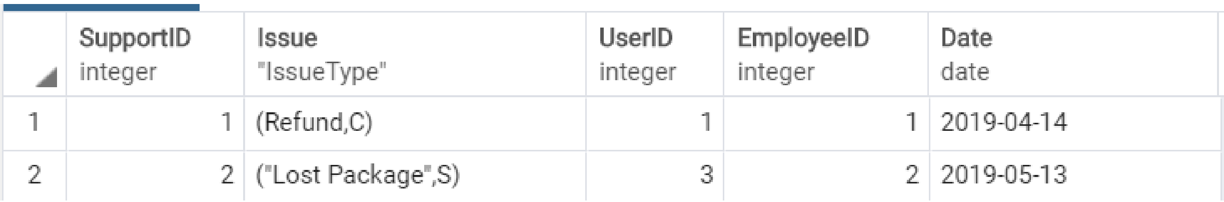
```
CREATE TABLE public."ReportsTo"
(
    "SubordinateID" integer NOT NULL,
    "SupervisorID" integer NOT NULL,
    CONSTRAINT "ReportsTo_pkey" PRIMARY KEY ("SubordinateID", "SupervisorID"),
    CONSTRAINT subordinate_fk FOREIGN KEY ("SubordinateID")
        REFERENCES public."Employee_Account" ("EmployeeID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT supervisor_fk FOREIGN KEY ("SupervisorID")
        REFERENCES public."Employee_Account" ("EmployeeID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
);
```



	SubordinateID integer	SupervisorID integer
1	2	1
2	3	1
3	2	3

SUPPORT: This table represents the relationship between a User_Account and Employee_Account in which a employee provides customer service to a user. The primary key is SupportID which allows the customer to keep track of any customer service they've received. Issue is a custom type which denotes what type of issue the customer had, like request for refund, request for exchange, or lost package. The date in which the customer request support is also recorded. This table references two other tables: EmployeeID identifies which employee helped the user and UserID

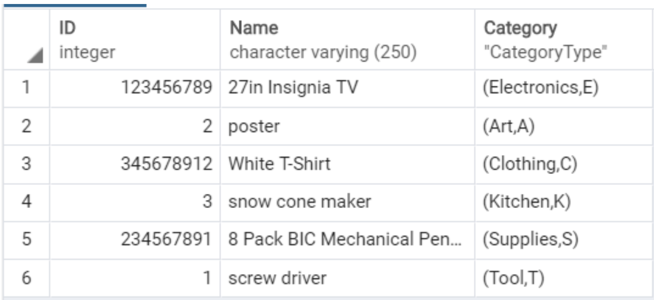
```
CREATE TABLE public."Support"
(
    "SupportID" integer NOT NULL,
    "Issue" "IssueType",
    "UserID" integer NOT NULL,
    "EmployeeID" integer NOT NULL,
    "Date" date NOT NULL,
    CONSTRAINT "Support_pkey" PRIMARY KEY ("SupportID"),
    CONSTRAINT employee_fk FOREIGN KEY ("EmployeeID")
        REFERENCES public."Employee_Account" ("EmployeeID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT user_fk FOREIGN KEY ("UserID")
        REFERENCES public."User_Account" ("UserId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```



	SupportID	Issue	UserID	EmployeeID	Date
	integer	"IssueType"	integer	integer	date
1	1	(Refund,C)	1	1	2019-04-14
2	2	("Lost Package",S)	3	2	2019-05-13

ITEM: This table will act as a catalogue of every item sold on JAM. Each item is identified with the primary key ID. Each entry will also have a name and the category of items it belongs to. CategoryType is a custom type that takes on values like electronics, clothing, furniture, etc. Many other relations will reference this table.

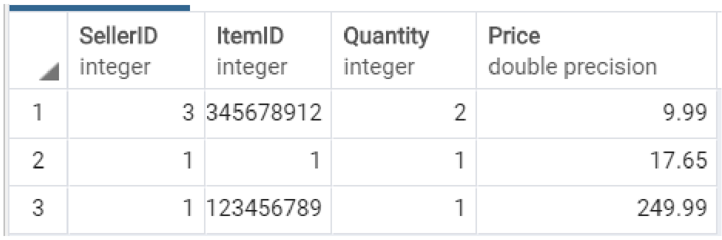
```
CREATE TABLE public."Item"
(
   "ID" integer NOT NULL,
   "Name" character varying NOT NULL,
   "Category" "CategoryType",
   CONSTRAINT "Item_pkey" PRIMARY KEY ("ID")
)
```



	ID	Name	Category
	integer	character varying (250)	"CategoryType"
1	123456789	27in Insignia TV	(Electronics,E)
2	2	poster	(Art,A)
3	345678912	White T-Shirt	(Clothing,C)
4	3	snow cone maker	(Kitchen,K)
5	234567891	8 Pack BIC Mechanical Pen...	(Supplies,S)
6	1	screw driver	(Tool,T)

SELLS: This table represents the relationship between a User_Account and an Item in which the user sells a Quantity of an Item at a Price. ItemID and SellerId form the primary key and are also foreign-keys to User_Account’s UserID and Item’s ID. The seller must sells items at a positive quantity and price.

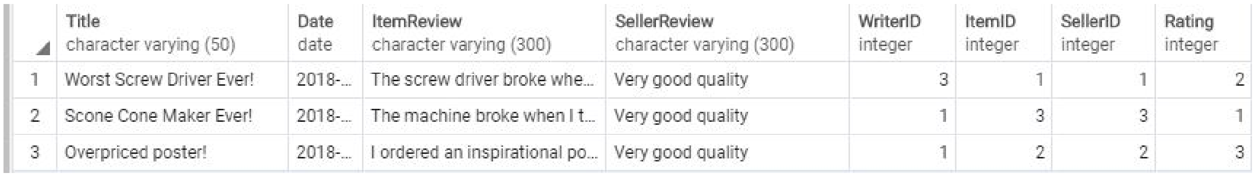
```
CREATE TABLE public."Sells"
(
   "SellerID" integer NOT NULL,
   "ItemID" integer NOT NULL,
   "Quantity" integer NOT NULL,
   "Price" double precision NOT NULL,
   CONSTRAINT sells_pk PRIMARY KEY ("ItemID", "SellerID")
   CONSTRAINT itemid_fk FOREIGN KEY ("ItemID")
      REFERENCES public."Item" ("ID") MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
   CONSTRAINT sellerid_fk FOREIGN KEY ("SellerID")
      REFERENCES public."User_Account" ("UserId") MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
   CONSTRAINT price_check CHECK ("Price" > 0::double precision) NOT VALID,
   CONSTRAINT quantity_check CHECK ("Quantity" > 0)
)
```



	SellerID integer	ItemID integer	Quantity integer	Price double precision
1	3	345678912	2	9.99
2	1	1	1	17.65
3	1	123456789	1	249.99

REVIEW: Review is a weak-entity type with an identifying relationship with Item. A review must have a title, date it was written, and a rating. The rating is based on the common 5-star rating system; a check constraint exists to make sure the rating is between 1 and 5, inclusively. The writer may also include a textual review of either the item itself or the seller. There are three foreign keys: one for the item, representing which Item it reviews; one for the seller, representing the User_Account that sells the item; and one for writer, representing the User_Account writing the review.

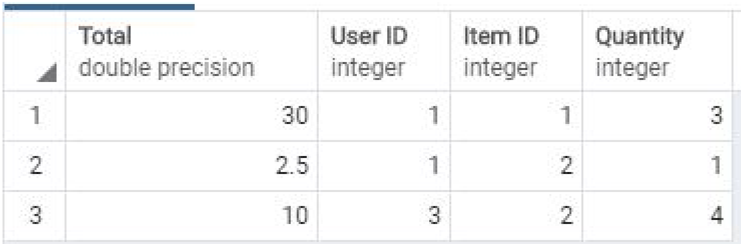
```
CREATE TABLE public."Review"
(
   "Title" character varying NOT NULL,
   "Date" date NOT NULL,
   "ItemReview" character varying,
   "SellerReview" character varying,
   "WriterID" integer NOT NULL,
   "ItemID" integer NOT NULL,
   "SellerID" integer NOT NULL,
   "Rating" integer NOT NULL,
   CONSTRAINT item_fk FOREIGN KEY ("ItemID")
      REFERENCES public."Item" ("ID") MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
   CONSTRAINT seller_fk FOREIGN KEY ("SellerID")
      REFERENCES public."User_Account" ("UserId") MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
   CONSTRAINT writer_fk FOREIGN KEY ("WriterID")
      REFERENCES public."User_Account" ("UserId") MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION,
   CONSTRAINT rating_check CHECK ("Rating" >= 1 AND "Rating" <= 5)
);
```



	Title character varying (50)	Date date	ItemReview character varying (300)	SellerReview character varying (300)	WriterID integer	ItemID integer	SellerID integer	Rating integer
1	Worst Screw Driver Ever!	2018-...	The screw driver broke whe...	Very good quality	3	1	1	2
2	Scone Cone Maker Ever!	2018-...	The machine broke when I t...	Very good quality	1	3	3	1
3	Overpriced poster!	2018-...	I ordered an inspirational po...	Very good quality	1	2	2	3

CART: This is a weak entity so it doesn't have a primary key. It has foreign keys User ID and Item ID that reference User ID and Item ID in tables User Account and Item, respectively. There is a check so a nonnegative number is placed for a quantity of a particular item and the total price of that particular items quantity. Ex) If a customer orders 10 copies of a textbook and each textbook costs $100, then quantity = 10, total = 1000.

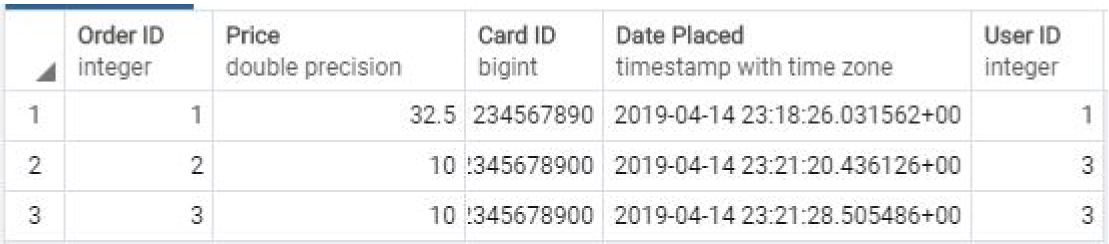
```
CREATE TABLE public."Cart"
(
    "Total" double precision,
    "User ID" integer NOT NULL,
    "Item ID" integer,
    "Quantity" integer,
    CONSTRAINT item_fk FOREIGN KEY ("Item ID")
        REFERENCES public."Item" ("ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT user_fk FOREIGN KEY ("User ID")
        REFERENCES public."User_Account" ("UserId") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT neg_num CHECK ("Total" > 0::double precision AND "Quantity" > 0)
);
```



	Total double precision	User ID integer	Item ID integer	Quantity integer
1	30	1	1	3
2	2.5	1	2	1
3	10	3	2	4

ORDER: Each order has a unique id called Order ID. All attributes in the Order table have the constraint NOT NULL because if an order is canceled the tuple representing the tuple doesn't remain it's just deleted. You can't have a NULL order. The order references a credit/debit card that the order is paid with. There is a check so a nonnegative number can’t be an entry for Price because an order can’t cost nothing or a negative amount.

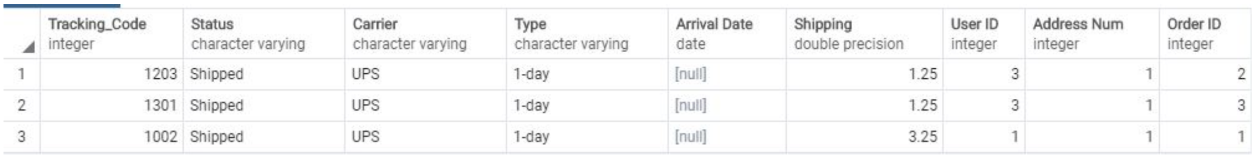
```
CREATE TABLE public."Order"
(
    "Order ID" integer NOT NULL,
    "Price" double precision NOT NULL,
    "Card ID" bigint NOT NULL,
    "Date Placed" timestamp with time zone NOT NULL,
    "User ID" integer NOT NULL,
    CONSTRAINT "Order_pkey" PRIMARY KEY ("Order ID"),
    CONSTRAINT card_fk FOREIGN KEY ("Card ID", "User ID")
        REFERENCES public."Card" ("Card_Number", "accountId") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT neg_price CHECK ("Price" > 0::double precision)
);
```



	Order ID integer	Price double precision	Card ID bigint	Date Placed timestamp with time zone	User ID integer
1	1	32.5	234567890	2019-04-14 23:18:26.031562+00	1
2	2	10	:345678900	2019-04-14 23:21:20.436126+00	3
3	3	10	:345678900	2019-04-14 23:21:28.505486+00	3

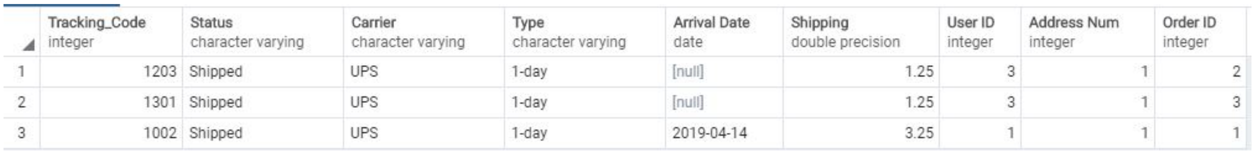
DELIVERY: Delivery is a weak entity. Each delivery has a unique tracking code. The only attribute that can be set to NULL is Arrival Date because if the delivery isn’t complete yet the delivery date is unknown. The Shipping attribute refers to the shipping and fees charge to the order that delivery references. There is a constraint on shipping to be >= 0 because you can have no shipping but you can’t have negative shipping. Delivery references an order and an address related to the unique order.

```
CREATE TABLE public."Delivery"
(
    "Tracking_Code" integer NOT NULL,
    "Status" character varying COLLATE pg_catalog."default" NOT NULL,
    "Carrier" character varying COLLATE pg_catalog."default" NOT NULL,
    "Type" character varying COLLATE pg_catalog."default" NOT NULL,
    "Arrival Date" date,
    "Shipping" double precision,
    "User ID" integer NOT NULL,
    "Address Num" integer NOT NULL,
    "Order ID" integer NOT NULL,
    CONSTRAINT delivery_unique UNIQUE ("Tracking_Code")
,
    CONSTRAINT address_fk FOREIGN KEY ("Address Num", "User ID")
        REFERENCES public."Address" ("Addr_num", "residentId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT order_fk FOREIGN KEY ("Order ID")
        REFERENCES public."Order" ("Order ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT neg_shipping CHECK ("Shipping" >= 0::double precision)
);
```



	Tracking_Code integer	Status character varying	Carrier character varying	Type character varying	Arrival Date date	Shipping double precision	User ID integer	Address Num integer	Order ID integer
1	1203	Shipped	UPS	1-day	[null]	1.25	3	1	2
2	1301	Shipped	UPS	1-day	[null]	1.25	3	1	3
3	1002	Shipped	UPS	1-day	[null]	3.25	1	1	1

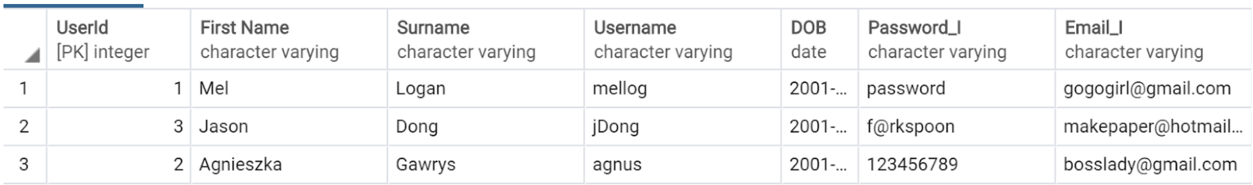
After delivery 1002 is delivered the database would be updated to this.



	Tracking_Code integer	Status character varying	Carrier character varying	Type character varying	Arrival Date date	Shipping double precision	User ID integer	Address Num integer	Order ID integer
1	1203	Shipped	UPS	1-day	[null]	1.25	3	1	2
2	1301	Shipped	UPS	1-day	[null]	1.25	3	1	3
3	1002	Shipped	UPS	1-day	2019-04-14	3.25	1	1	1

USER ACCOUNT: User Account refers to any user on the application, whether it be an employee or buyer. It is a child of Account. Each user has a first name, surname, date of birth, and username. These must be NOT NULL but do not have to be unique since users may have the same name. UserId will be a UNIQUE and NOT NULL attribute as the primary key, which will serve the purpose of identifying a specific user.

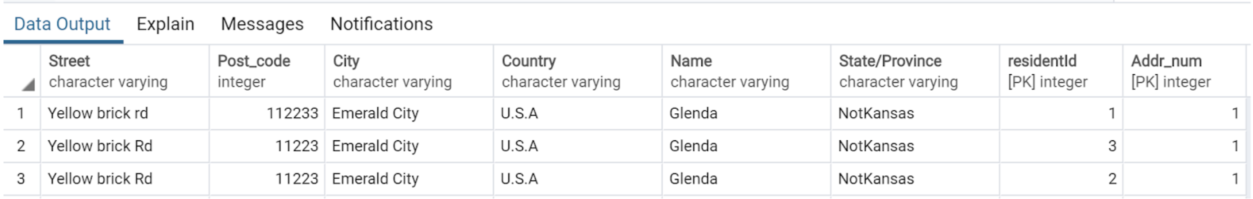
```
CREATE TABLE public."User_Account"
(
    "UserId" integer NOT NULL,
    "First Name" character varying COLLATE pg_catalog."default" NOT NULL,
    "Surname" character varying COLLATE pg_catalog."default" NOT NULL,
    "Username" character varying COLLATE pg_catalog."default" NOT NULL,
    "DOB" date NOT NULL,
    "Password_I" character varying COLLATE pg_catalog."default" NOT NULL,
    "Email_I" character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT user_key PRIMARY KEY ("UserId")
)
```



	UserId [PK] integer	First Name character varying	Surname character varying	Username character varying	DOB date	Password_I character varying	Email_I character varying
1	1	Mel	Logan	mellog	2001-…	password	gogogirl@gmail.com
2	3	Jason	Dong	jDong	2001-…	f@rkspoon	makepaper@hotmail…
3	2	Agnieszka	Gawrys	agnus	2001-…	123456789	bosslady@gmail.com

ADDRESS: The Address stores the billing address and/or the shipping address for a given order. It contains a foreign key to the UserId, which is a unique value. The same address may be used by different users and so it does not have to be UNIQUE, but an address must exist for a shipment and billing, hence the values may not be NULL. There is an address Id which other relations will reference.

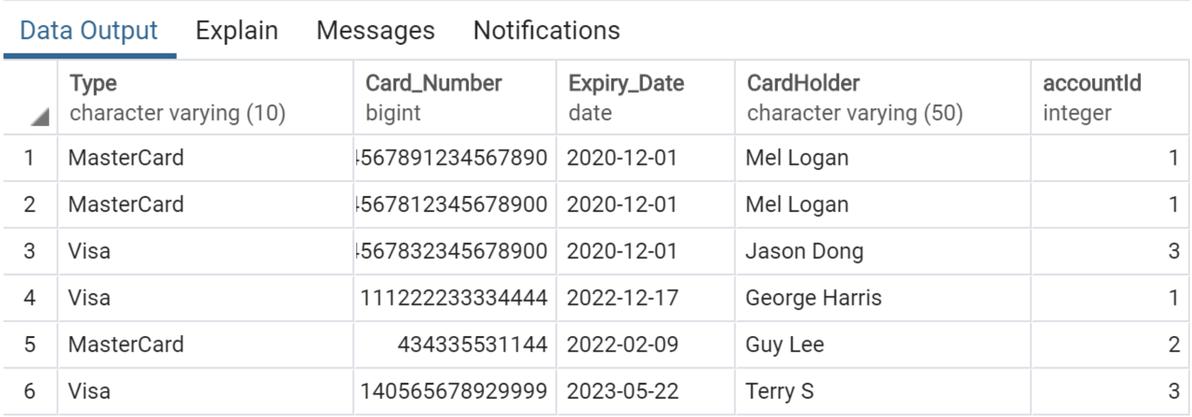
```
CREATE TABLE public."Address"
(
   "Street" character varying COLLATE pg_catalog."default" NOT NULL,
   "Post_code" integer NOT NULL,
   "City" character varying COLLATE pg_catalog."default" NOT NULL,
   "Country" character varying COLLATE pg_catalog."default" NOT NULL,
   "Name" character varying COLLATE pg_catalog."default" NOT NULL,
   "State/Province" character varying COLLATE pg_catalog."default" NOT NULL,
   "residentId" integer NOT NULL,
   "Addr_num" integer NOT NULL,
   CONSTRAINT address_key PRIMARY KEY ("Addr_num", "residentId"),
   CONSTRAINT account_identification FOREIGN KEY ("residentId")
      REFERENCES public."User_Account" ("UserId") MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE CASCADE
)
```



	Street character varying	Post_code integer	City character varying	Country character varying	Name character varying	State/Province character varying	residentId [PK] integer	Addr_num [PK] integer
1	Yellow brick rd	112233	Emerald City	U.S.A	Glenda	NotKansas	1	1
2	Yellow brick Rd	11223	Emerald City	U.S.A	Glenda	NotKansas	3	1
3	Yellow brick Rd	11223	Emerald City	U.S.A	Glenda	NotKansas	2	1

CARD: This entity represents any card the user wants to pay with. It is a weak entity since it may not exist on the application without a pre-existing user. Thus, it has a foreign key to the UserId, a unique value from the User_Account table. The Card_Number may appear in multiple accounts but the combination of the card number and account ID must be UNIQUE and NOT NULL. The rest of the attributes, Card_Holder, Expiry_Date, Type, are NOT NULL but do not have to be unique.

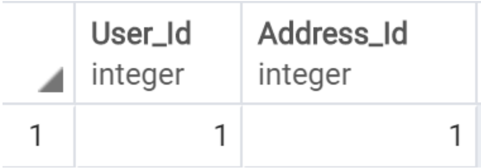
```
CREATE TABLE public."Card"
(
   "Type" character varying(10) COLLATE pg_catalog."default" NOT NULL,
   "Card_Number" bigint NOT NULL,
   "Expiry_Date" date NOT NULL,
   "CardHolder" character varying(50) COLLATE pg_catalog."default" NOT NULL,
   "accountId" integer NOT NULL,
   CONSTRAINT "Card+Account" UNIQUE ("Card_Number", "accountId")
   CONSTRAINT "User_Identification" FOREIGN KEY ("accountId")
      REFERENCES public."User_Account" ("UserId") MATCH SIMPLE
      ON UPDATE NO ACTION
      ON DELETE NO ACTION
)
```



	Type character varying (10)	Card_Number bigint	Expiry_Date date	CardHolder character varying (50)	accountId integer
1	MasterCard	567891234567890	2020-12-01	Mel Logan	1
2	MasterCard	567812345678900	2020-12-01	Mel Logan	1
3	Visa	567832345678900	2020-12-01	Jason Dong	3
4	Visa	111222233334444	2022-12-17	George Harris	1
5	MasterCard	434335531144	2022-02-09	Guy Lee	2
6	Visa	140565678929999	2023-05-22	Terry S	3

BILLING: Billing table serves to link the payment with the address. It consists of two foreign keys, a UNIQUE card number from Card Table, and a UNIQUE address Id from the address table.

```
CREATE TABLE public."Billing"
(
   "User_Id" integer NOT NULL,
   "User_Id" integer NOT NULL,
   "Address_Id" integer NOT NULL,
    CONSTRAINT "AddressUserId" FOREIGN KEY ("Address_Id", "User_Id")
        REFERENCES public."Address" ("Addr_num", "residentId") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```



	User_Id integer	Address_Id integer
1	1	1