Codd Movie Rentals

Database Management

April 19, 2016

Zachary Fong

INSERT TABLE OF CONTENTS HERE

Table of Contents

Executive Summary 3

Entity Relationship Diagram 4

Tables 5

Conditions 6

Genre 7

Actors 8

Movies 9

Rent_Status 10

Cust_rentals 11

Payment 12

Views 13

PayStatus 14

CustomerRoster 15

MissingMovie 16

Reports 17

Total Income 18

Average Income 19

Triggers 20

Security 21

Store Procedures 22

Known Problems 23

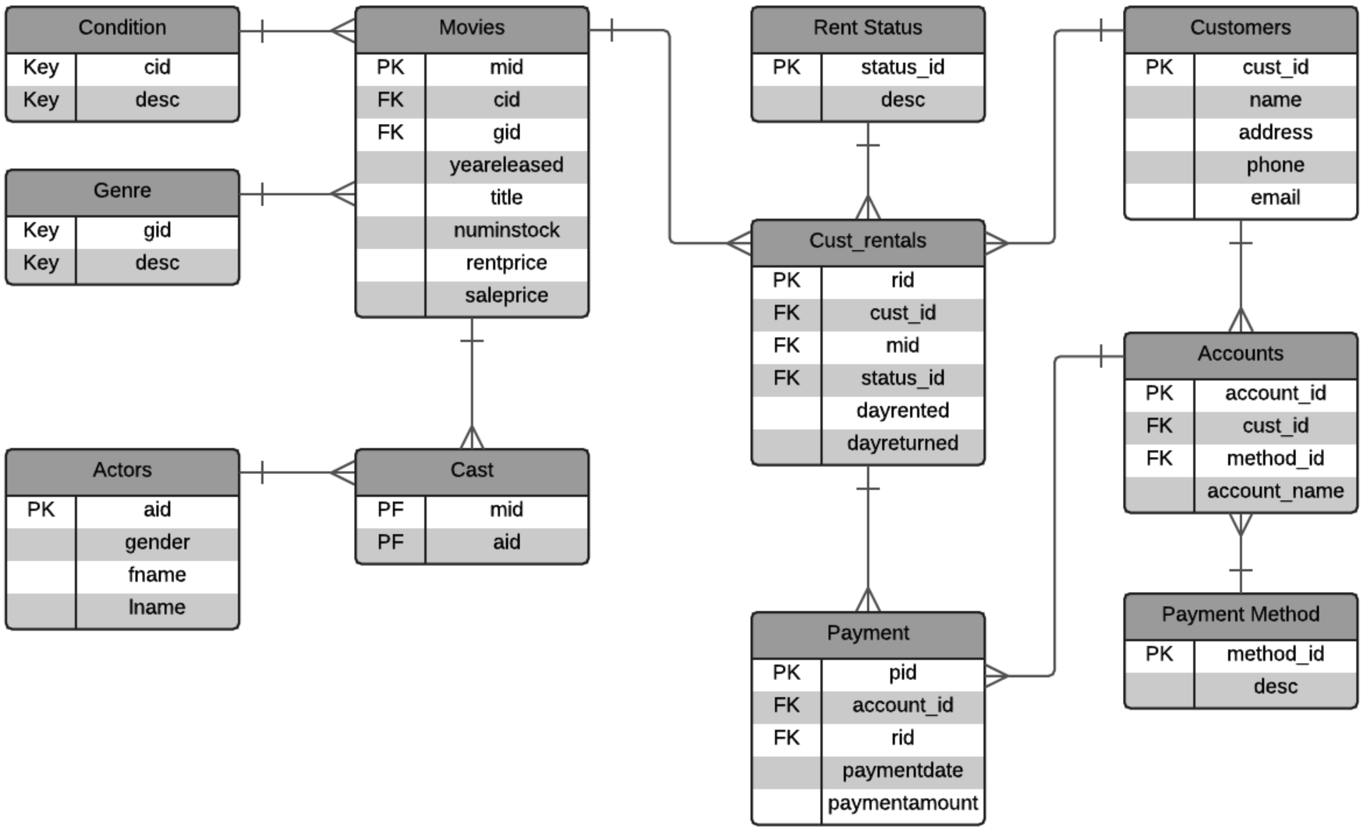
Future Enhancements 23

Executive Summary

The purpose of this document is to outline a database to record resources within a movie rental store. Specifically, this database records the condition of a move, its genre, title of the movie, its price, and customer information. This will allow the manager of the store to keep track of different movies and whether or not a customer returned the movie in a worse condition or if they even returned it at all.

This document will provide an overview of the database. This includes the various tables in the database, the purpose of each table, and their functional dependencies, and more.

ENTITY RELATIONSHIP DIAGRAM



TABLES

CONDITION

Purpose

This table is used to store the state of the conditions of different movies

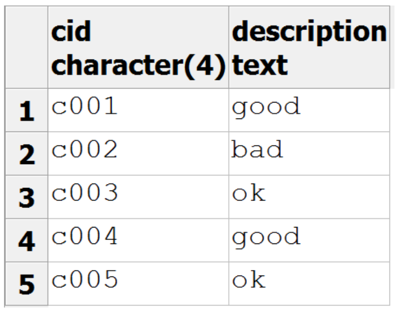
Create Statement

```
CREATE TABLE Condition (
            cid char(4) not null,
            description text,
PRIMARY KEY (cid)
);
```

Functional Dependencies

cid→ desc

Sample Data



	cid character(4)	description text
1	c001	good
2	c002	bad
3	c003	ok
4	c004	good
5	c005	ok

Purpose

This table is used to identify the genres for each movie.

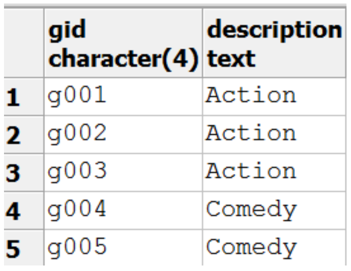
Create Statement

```
CREATE TABLE Genre (
        gid char(4) not null,
        description text,
PRIMARY KEY (gid)
);
```

Functional Dependencies

gid → desc

Sample Data



	gid character(4)	description text
1	g001	Action
2	g002	Action
3	g003	Action
4	g004	Comedy
5	g005	Comedy

ACTORS

Purpose

This table is used to store a list of actor so a user can search for movies by actors.

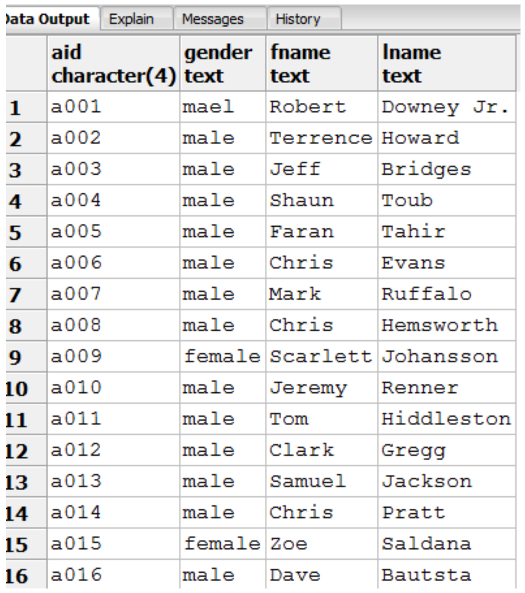
Create Statement

```
CREATE TABLE Actors (
        aid char(4) not null,
        gender text,
        fname text,
        lname text,
Primary Key (aid)
);
```

Functional Dependencies

aid → gender, fname, lname

Sample Data



	aid character(4)	gender text	fname text	lname text
1	a001	mael	Robert	Downey Jr.
2	a002	male	Terrence	Howard
3	a003	male	Jeff	Bridges
4	a004	male	Shaun	Toub
5	a005	male	Faran	Tahir
6	a006	male	Chris	Evans
7	a007	male	Mark	Ruffalo
8	a008	male	Chris	Hemsworth
9	a009	female	Scarlett	Johansson
10	a010	male	Jeremy	Renner
11	a011	male	Tom	Hiddleston
12	a012	male	Clark	Gregg
13	a013	male	Samuel	Jackson
14	a014	male	Chris	Pratt
15	a015	female	Zoe	Saldana
16	a016	male	Dave	Bautsta

MOVIES

Purpose

This table is used to store the different movies available for rent.

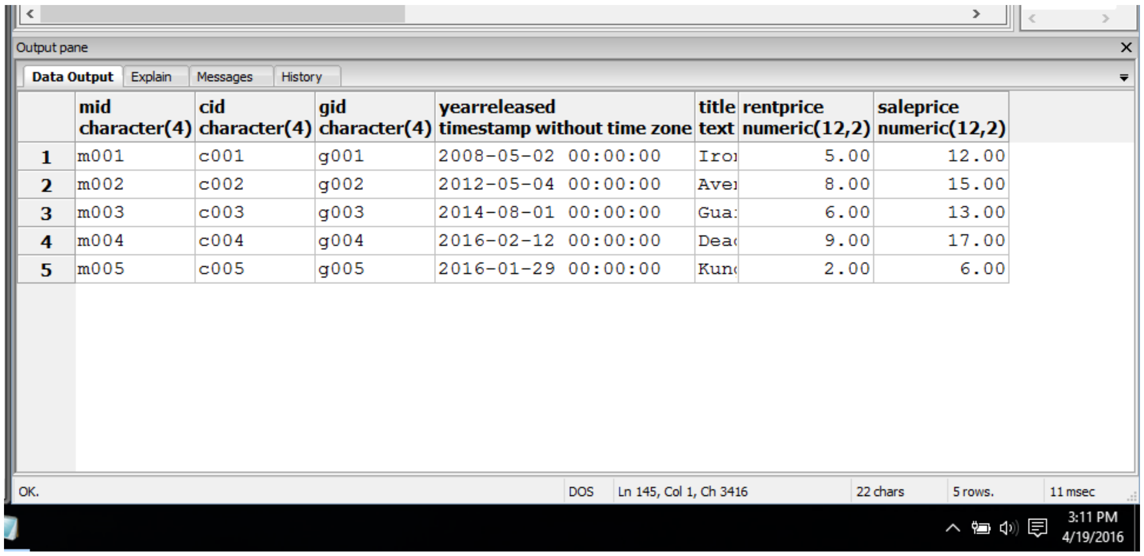
Create Statement

```
CREATE TABLE Movies (
        mid char(4) not null,
        cid char(4) not null references Condition(cid),
        gid char(4) not null references Genre(gid),
        yearrelased int,
        title text,
        rentprice int,
        saleprice int,
PRIMARY KEY (mid)
);
```

Functional Dependencies

mid→ cid, gid, yearreleased, title, rentprice, saleprice

Sample Data



Output pane

Data Output | Explain | Messages | History

	mid character(4)	cid character(4)	gid character(4)	yearreleased timestamp without time zone	title text	rentprice numeric(12,2)	saleprice numeric(12,2)
1	m001	c001	g001	2008-05-02 00:00:00	Iro	5.00	12.00
2	m002	c002	g002	2012-05-04 00:00:00	Ave	8.00	15.00
3	m003	c003	g003	2014-08-01 00:00:00	Gua:	6.00	13.00
4	m004	c004	g004	2016-02-12 00:00:00	Dea	9.00	17.00
5	m005	c005	g005	2016-01-29 00:00:00	Kun	2.00	6.00

OK. DOS Ln 145, Col 1, Ch 3416 22 chars 5 rows. 11 msec

3:11 PM 4/19/2016

RENT_STATUS

Purpose

This table is used to keep track of the status of rented movies.

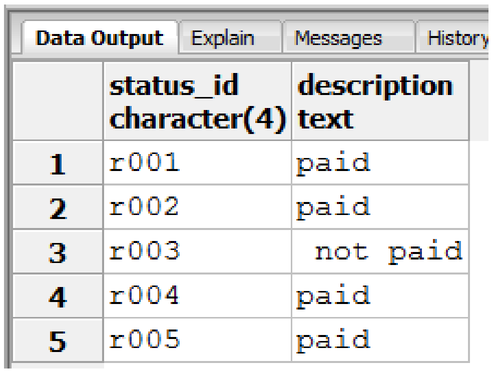
Create Statement

```
CREATE TABLE RentStatus (
        status_id char(4) not null,
        description text,
PRIMARY KEY(status_id)
);
```

Functional Dependencies

status_id→desc

Sample Data



	status_id character(4)	description text
1	r001	paid
2	r002	paid
3	r003	not paid
4	r004	paid
5	r005	paid

CUST_RENTALS

Purpose

This table is used to keep track of when a customer rented a movie and whether or not they returned it.

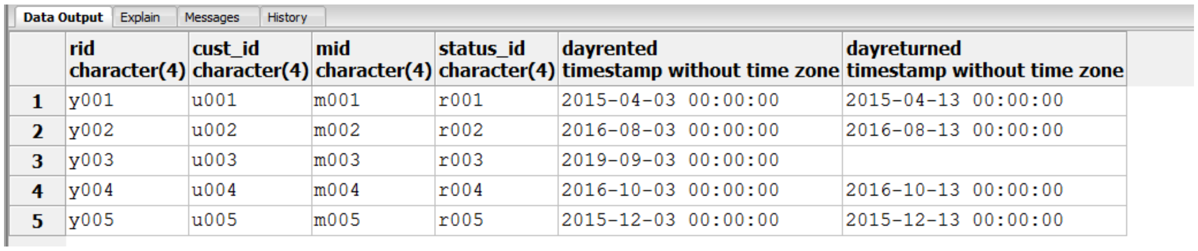
Create Statement

```
CREATE TABLE Cust_rentals (
        rid char(4) not null,
        cust_id char(4) not null references Customers(cust_id),
        mid char(4) not null references Movies(mid),
        status_id char(4) not null references RentStatus(status_id),
        dayrented timestamp not null,
        dayreturned timestamp,
PRIMARY KEY(rid)
);
```

Functional Dependencies

rid→cust_id, mid, status_id, dayrented, dayreturned

Sample Data



Data Output | Explain | Messages | History

	rid character(4)	cust_id character(4)	mid character(4)	status_id character(4)	dayrented timestamp without time zone	dayreturned timestamp without time zone
1	y001	u001	m001	r001	2015-04-03 00:00:00	2015-04-13 00:00:00
2	y002	u002	m002	r002	2016-08-03 00:00:00	2016-08-13 00:00:00
3	y003	u003	m003	r003	2019-09-03 00:00:00	
4	y004	u004	m004	r004	2016-10-03 00:00:00	2016-10-13 00:00:00
5	y005	u005	m005	r005	2015-12-03 00:00:00	2015-12-13 00:00:00

PAYMENT

Purpose

This table is used to track when the customer paid and how much they paid.

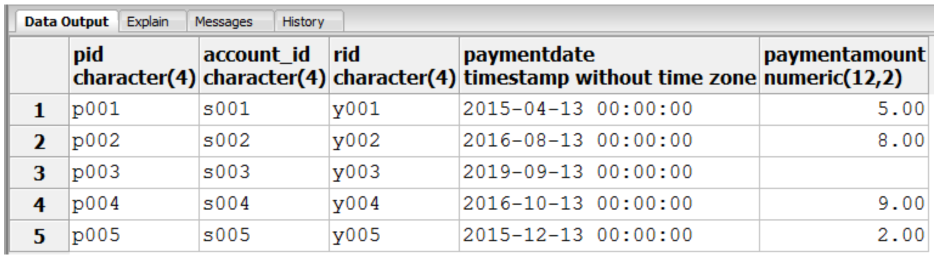
Create Statement

```
CREATE TABLE Payment (
        pid char(4) not null,
        account_id char(4) not null references Accounts(account_id),
        rid char(4) not null references Cust_rentals(rid),
        paymentdate timestamp,
        paymentamount decimal (12,2),
PRIMARY KEY (pid)
);
```

Functional Dependencies

pid→account_id, rid, paymentdate, paymentamount

Sample Data



Data Output | Explain | Messages | History

	pid character(4)	account_id character(4)	rid character(4)	paymentdate timestamp without time zone	paymentamount numeric(12,2)
1	p001	s001	y001	2015-04-13 00:00:00	5.00
2	p002	s002	y002	2016-08-13 00:00:00	8.00
3	p003	s003	y003	2019-09-13 00:00:00	
4	p004	s004	y004	2016-10-13 00:00:00	9.00
5	p005	s005	y005	2015-12-13 00:00:00	2.00

CUSTOMERS

Purpose

This table is used to store the customer's information.

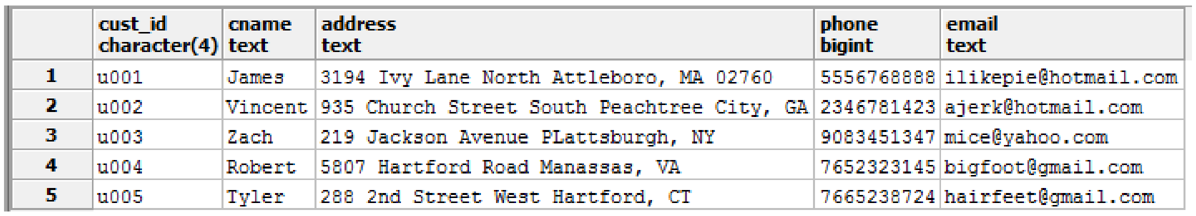
Create Statement

```
CREATE TABLE Customers (
        cust_id char(4) not null,
        cname text,
        address text,
        phone bigint not null,
        email text,
PRIMARY KEY (cust_id)
);
```

Functional Dependencies

cust_id→name, address, phone, email

Sample Data



	cust_id character(4)	cname text	address text	phone bigint	email text
1	u001	James	3194 Ivy Lane North Attleboro, MA 02760	5556768888	ilikepie@hotmail.com
2	u002	Vincent	935 Church Street South Peachtree City, GA	2346781423	ajerk@hotmail.com
3	u003	Zach	219 Jackson Avenue PLattsburgh, NY	9083451347	mice@yahoo.com
4	u004	Robert	5807 Hartford Road Manassas, VA	7652323145	bigfoot@gmail.com
5	u005	Tyler	288 2nd Street West Hartford, CT	7665238724	hairfeet@gmail.com

ACCOUNTS

Purpose

This table is used to keep a record of the accounts of customers.

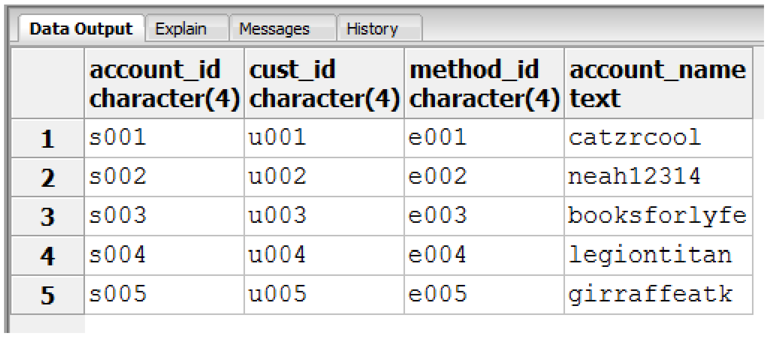
Create Statement

```
CREATE TABLE Accounts (
        account_id char(4) not null,
        cust_id char(4) not null references Customers(cust_id),
        method_id char(4) not null references PaymentMethod(method_id),
        account_name text,
PRIMARY KEY (account_id)
);
```

Functional Dependencies

account_id→cust_id, method_id, account_name

Sample Data



	account_id character(4)	cust_id character(4)	method_id character(4)	account_name text
1	s001	u001	e001	catzrcool
2	s002	u002	e002	neah12314
3	s003	u003	e003	booksforlyfe
4	s004	u004	e004	legiontitan
5	s005	u005	e005	girraffeatk

PAYMENT METHOD

Purpose

This table is used to store the methods customers used to pay.

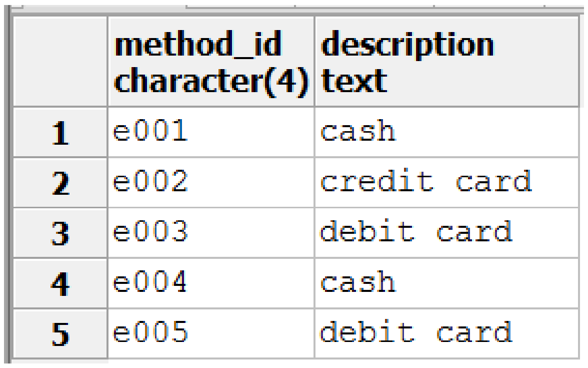
Create Statement

```
CREATE TABLE PaymentMethod (
        method_id char(4) not null,
        description text,
PRIMARY KEY (method_id)
);
```

Functional Dependencies

method_id→desc

Sample Data



	method_id character(4)	description text
1	e001	cash
2	e002	credit card
3	e003	debit card
4	e004	cash
5	e005	debit card

VIEWS

PayStatus

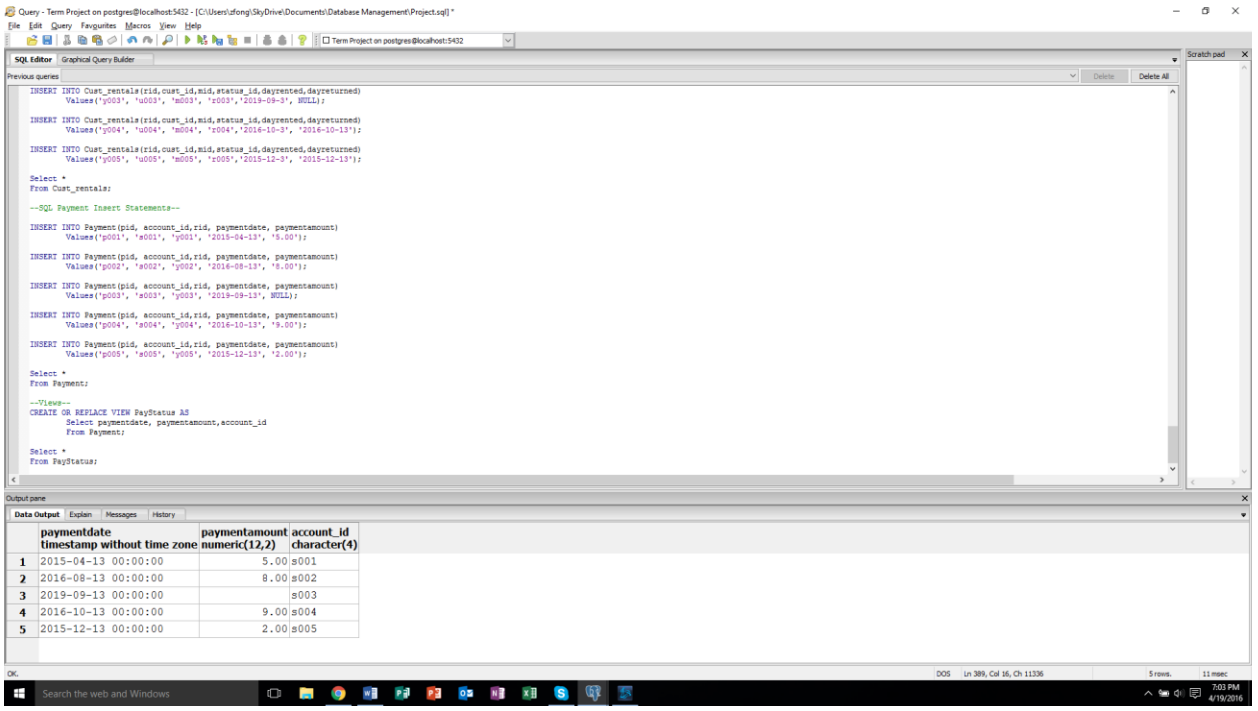
Purpose

This view is used to determine the payment status of a movie once it is returned and what they paid with.

Create Statement

CREATE VIEW PayStatus AS
    Select paymentdate, paymentamount
    From Payment;

Sample Data



Query - Term Project on postgres@localhost:5432 - [C:\Users\zfong\SkyDrive\Documents\Database Management\Project.sql] *

```
INSERT INTO Cust_rentals(rid,cust_id,mid,status_id,dayrented,dayreturned)
        Values('y003', 'u003', 'm003', 'r003','2019-09-3', NULL);

INSERT INTO Cust_rentals(rid,cust_id,mid,status_id,dayrented,dayreturned)
        Values('y004', 'u004', 'm004', 'r004','2016-10-3', '2016-10-13');

INSERT INTO Cust_rentals(rid,cust_id,mid,status_id,dayrented,dayreturned)
        Values('y005', 'u005', 'm005', 'r005','2015-12-3', '2015-12-13');

Select *
From Cust_rentals;

--SQL Payment Insert Statements--

INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p001', 's001', 'y001', '2015-04-13', '5.00');

INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p002', 's002', 'y002', '2016-08-13', '8.00');

INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p003', 's003', 'y003', '2019-09-13', NULL);

INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p004', 's004', 'y004', '2016-10-13', '9.00');

INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p005', 's005', 'y005', '2015-12-13', '2.00');

Select *
From Payment;

--Views--
CREATE OR REPLACE VIEW PayStatus AS
        Select paymentdate, paymentamount,account_id
        From Payment;

Select *
From PayStatus;
```

	paymentdate timestamp without time zone	paymentamount numeric(12,2)	account_id character(4)
1	2015-04-13 00:00:00	5.00	s001
2	2016-08-13 00:00:00	8.00	s002
3	2019-09-13 00:00:00		s003
4	2016-10-13 00:00:00	9.00	s004
5	2015-12-13 00:00:00	2.00	s005

OK. DOS Ln 389, Col 16, Ch 11336 5 rows. 11 msec

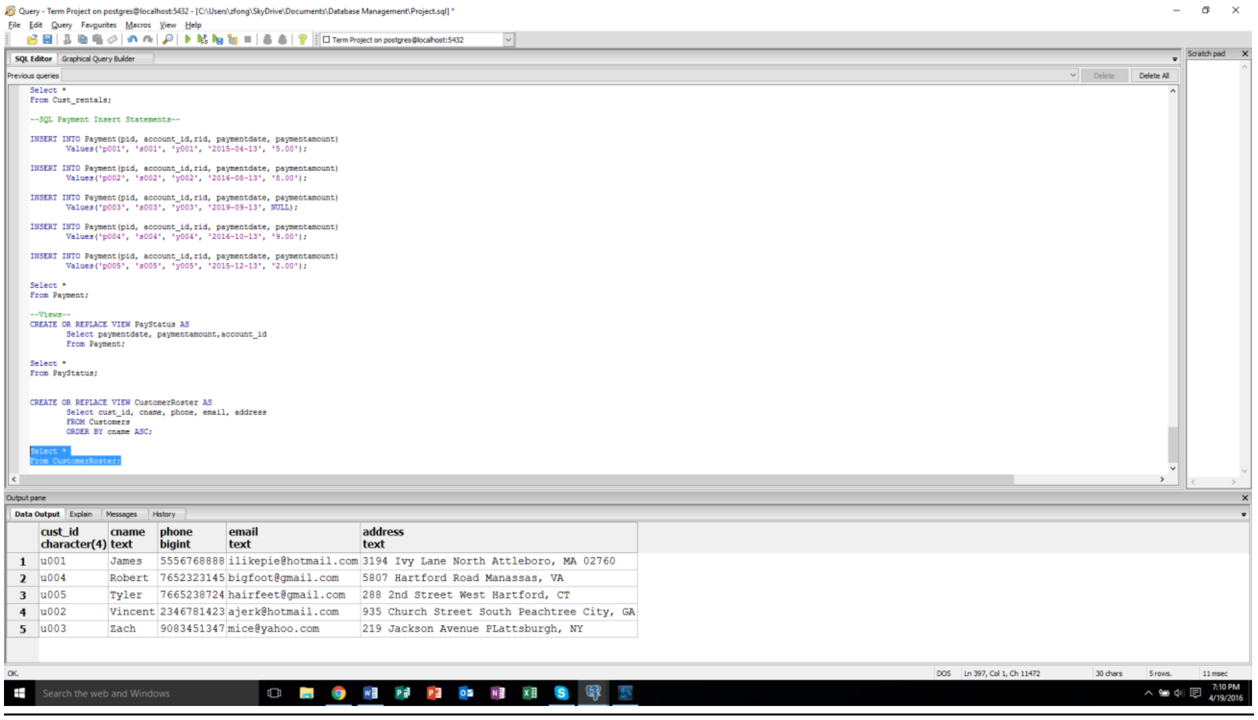
Purpose

This view shows the entire list of customers and all their relevant information.

Create Statement

```
CREATE VIEW CustomerRoster AS
        Select c.cid AS Customer ID,
               lname, fname,
               Phone, email,
               Address,
        From Customers c
        Order By lname ASC;
```

Sample Data



```
Select *
From Cust_rentals;

--SQL Payment Insert Statements--

INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p001', 's001', 'y001', '2015-04-13', '5.00');

INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p002', 's002', 'y002', '2016-08-13', '8.00');

INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p003', 's003', 'y003', '2019-09-13', NULL);

INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p004', 's004', 'y004', '2016-10-13', '9.00');

INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p005', 's005', 'y005', '2015-12-13', '2.00');

Select *
From Payment;

--Views--
CREATE OR REPLACE VIEW PayStatus AS
        Select paymentdate, paymentamount,account_id
        From Payment;

Select *
From PayStatus;


CREATE OR REPLACE VIEW CustomerRoster AS
        Select cust_id, cname, phone, email, address
        FROM Customers
        ORDER BY cname ASC;

Select *
From CustomerRoster;
```

	cust_id character(4)	cname text	phone bigint	email text	address text
1	u001	James	5556768888	ilikepie@hotmail.com	3194 Ivy Lane North Attleboro, MA 02760
2	u004	Robert	7652323145	bigfoot@gmail.com	5807 Hartford Road Manassas, VA
3	u005	Tyler	7665238724	hairfeet@gmail.com	288 2nd Street West Hartford, CT
4	u002	Vincent	2346781423	ajerk@hotmail.com	935 Church Street South Peachtree City, GA
5	u003	Zach	9083451347	mice@yahoo.com	219 Jackson Avenue PLattsburgh, NY

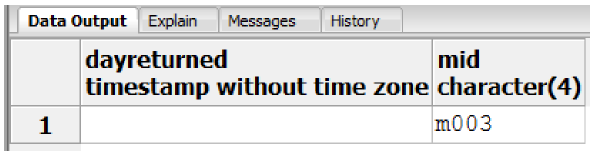
Purpose

The purpose of this view is to check whether any customers have not returned their movies.

Create Statement

```
CREATE VIEW MissingMovies  AS
        Select dayreturned, mid
        From Cust_rentals
        Where dayreturned IS NULL;
```

Sample Data



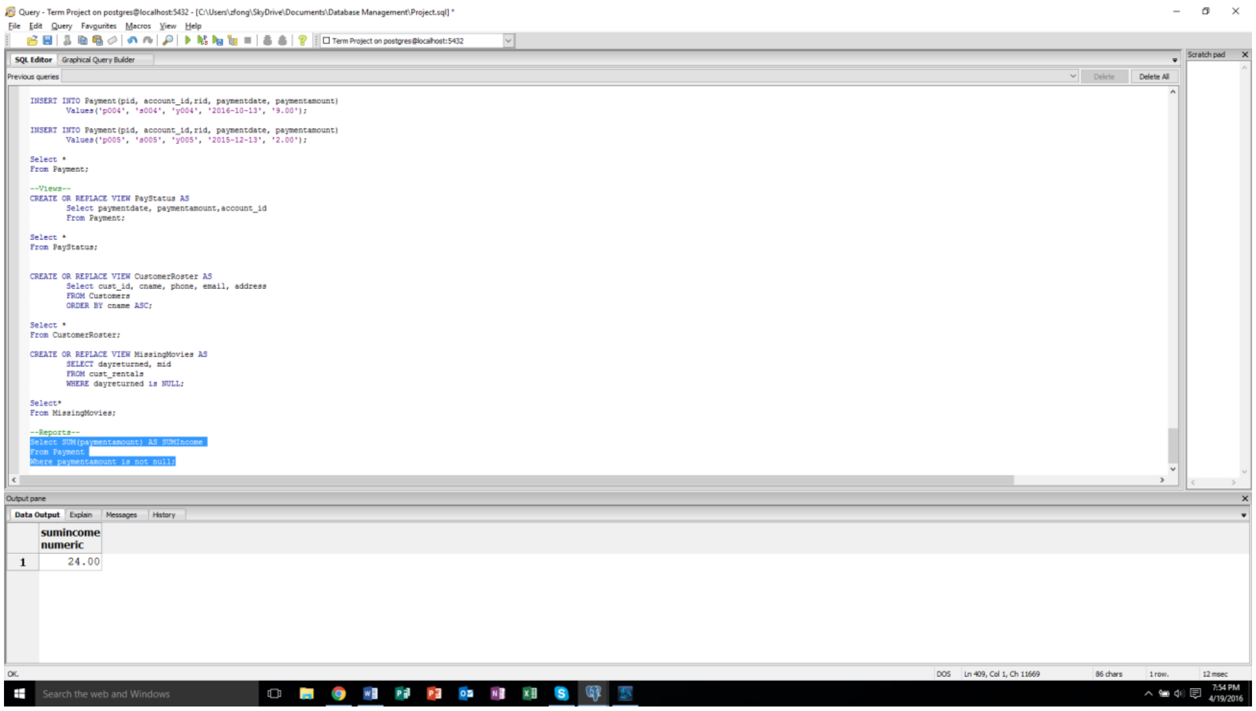
	dayreturned timestamp without time zone	mid character(4)
1		m003

Reports

Total Income

This is used to determine the total income of all movies that have either been rented or have been completely bought.

Query
Select SUM(paymentamount) AS SUMIncome
From Payment
Where paymentamount is not null;


Query - Term Project on postgres@localhost:5432 - [C:\Users\zfong\SkyDrive\Documents\Database Management\Project.sql] *
File Edit Query Favourites Macros View Help
Term Project on postgres@localhost:5432
SQL Editor
Graphical Query Builder
Previous queries
Delete
Delete All
Scratch pad
Output pane
Data Output
Explain
Messages
History
OK.
DOS
Ln 409, Col 1, Ch 11669
86 chars
1 row.
12 msec
Search the web and Windows
7:54 PM
4/19/2016

```
INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p004', 's004', 'y004', '2016-10-13', '9.00');

INSERT INTO Payment(pid, account_id,rid, paymentdate, paymentamount)
        Values('p005', 's005', 'y005', '2015-12-13', '2.00');

Select *
From Payment;

--Views--
CREATE OR REPLACE VIEW PayStatus AS
        Select paymentdate, paymentamount,account_id
        From Payment;

Select *
From PayStatus;


CREATE OR REPLACE VIEW CustomerRoster AS
        Select cust_id, cname, phone, email, address
        FROM Customers
        ORDER BY cname ASC;

Select *
From CustomerRoster;

CREATE OR REPLACE VIEW MissingMovies AS
        SELECT dayreturned, mid
        FROM cust_rentals
        WHERE dayreturned is NULL;

Select*
From MissingMovies;

--Reports--
Select SUM(paymentamount) AS SUMIncome
From Payment
Where paymentamount is not null;
```

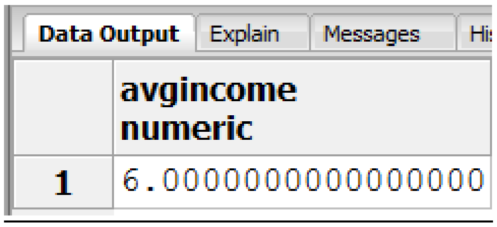
```
  | sumincome
  | numeric
1 |     24.00
```

Average Income

This is used for the manager to be able to see what the average income is for all returned rented movies and movie sales.

<u>Query</u>

Select AVG(paymentamount) AS AVGIncome
From Payment
Where paymentamount is not null;



```
   avgincome
   numeric
1  6.0000000000000000
```

Triggers

UnpaidRental

Purpose

When a customer has either forgotten to pay for their rented movie or has loaned the movie for an extended time period the table will immediately increase the amount due whenever the customer decides to return the movie

Query

```
CREATE TRIGGER UnpaidRental
AFTER UPDATE ON Payment
        FOR EACH ROW EXECUETE PROCEDURE addpayment();
```

Security

Database Administrator

The Database administrator has access to everything.

GRANT ALL PRIVELEGES ON ALL TABLES IN SCHEMA public TO dbAdministator;

Manager

Manager is able to see all information in the database with the exception of not being able to see the customer's financial information.

```
GRANT SELECT ON Movies TO manager;
GRANT SELECT, UPDATE ON Rent Status TO manager;
GRANT SELECT, INSERT, UPDATE ON Customers TO manager;
GRANT SELECT, INSERT, UPDATE ON Accounts TO manager;
GRANT SELECT, UPDATE Actors TO manager;
GRANT SELECT, INSERT, UPDATE ON Genre TO manager;
GRANT SELECT, INSERT, UPDATE ON Cust_rentals TO manager;
GRANT SELECT, INSERT, UPDATE ON Condition TO manager;
```

Store Procedures

<u>Adding New Movies</u>

The purpose of this is whenever a new movie is released the database administrator can add the movie to the database.

<u>Query</u>

```
CREATE OR REPLACE FUNCTION addmovie(char, char, char, timestamp, text, decimal, decimal)
returns refcursor AS
$$

DECLARE
        Vchar           char:= $1
        Vchar           char:= $2
        Vchar           char:= $3
        Vtimestamp      timestamp:= $4
        Vtext           text:= $5
        Vdec            decimal:= $6
        Vdec            decimal:= $7
        Resultset       refcursor:= null

BEGIN
        INSERT INTO movies(mid, cid, gid, yearreleased, title, rentprice, saleprice)
        VALUES (vchar, vchar, vchar, vtimestamp, vtext, vdec, vdec)
        Return resultset
END;
$$
LANGUAGE plpgsql
```

Known Problems

- Did not account for actors being in multiple movies examples would be Robert Downey Jr being in multiple sequels of Iron Man and in the Avengers.
- Did not implement a way for customers to search for movies based on who directed the movie
- Some movies have multiple genres did not include all genres the movie may apply to only the main genre.

Future Enhancements

- Allow an employment table to keep track of an employee roster and see when they clock in and out
- The employment table should be separated from managerial positions and regular employees.