Constantine Macris (252850)

December 9th, 2021

ITC 3160: Fundamentals of RDBMS

Dr. Maira Kotsovoulou

DATABASE APPLICATION

Telecom Field Service

Contents

[Business Description 2](#_Toc89982107)

[Business Requirements 2](#_Toc89982108)

[Logical Entity Relationship Diagram 2](#_Toc89982109)

[Description of relationships and constraints: 3](#_Toc89982110)

[Physical Entity Relationship Diagram 4](#_Toc89982111)

[Create Table Statements 4](#_Toc89982112)

[Source Code for Triggers 8](#_Toc89982113)

[Insert Statements and Update Statement 9](#_Toc89982114)

[Select Statements and Results 13](#_Toc89982115)

[Applications Screenshots and Descriptions 15](#_Toc89982116)

# Company Description

The business of choice for this project is Triturion LLC—a telecom field service business. This company provides critical IT field service and low voltage system support throughout the greater Chicago area. The head field service provider operates his business independently of any other major field service companies. Rather, has a reviewed online profile through which he receives job offers via various recruiting websites/applications. This means that he must manage his own schedule (choosing from available job opportunities), resources, and any subcontractors. To keep track of all this data in an organized fashion, I was recruited to design and implement a database application.

# Business Requirements

To keep all the data related—but also separate and organized—many different data categories must be utilized. The key data categories to be tracked are the following:

* Customers and their pertinent information: fundamental data necessary for planning, reimbursements, and valuable working relationships
* Job scheduling with location, distance, drive times, and estimated work time onsite: these data are necessary to maximize work planning efficiency
* Labelled expenses: needed to keep track of reimbursables and to make profit calculations easier and more accurate
* Inventory: keeping track of depletable and non-depletable assets helps plan for jobs and avoid scheduling delays
* Pricelist: useful to be able to give a quick, rough estimate job quote to potential customers
* Subcontractors and their pertinent information: having a list of subcontractors with estimated time until availability and rates can help with managing work overload
* Subcontracts: this is needed to keep track of which contractors are assigned to which jobs

# Logical Entity Relationship Diagram

Diagram

Description automatically generated

## Description of relationships and constraints:

The job entity can be seen as the “hub” of this relational model, as it ties together much key information and will likely be referenced very often. The job entity has a many-to-many relationship with subcontractors. This means that a single job can have multiple subcontractors, and a single subcontractor can be assigned to multiple jobs. When the logical diagram is engineered to a relational diagram, a new table describing this many-to-many relationship will be created. I have named that new table ‘subcontract’, and it provides the links between jobs and subcontractors. This relationship is also mutually optional, which means that there can be subcontractors with no jobs, and jobs with no subcontractors.

The customer entity is another key entity which stores much vital information. It has a one-to-many relationship with the job table, and the customer’s end of the relationship is optional—this means a customer can be linked to any number of jobs (including zero), while each job *must* have 1 single customer for reference.

Finally, the inventory, job, and customer table are all related to the expense table through an optional one-to-many relationship. What this means is that each single inventory item, job, and customer can, *independently,* create any number of discrete expenses. For example, a complex inventory item, such as a drill set, may incur a purchase cost at one point, and a repair cost at another point; a job could evidently incur multiple separate expenses such as gas, cabling, or other various network hardware components; and a customer can independently incur expenses if they have anything done for them that does not qualify as a job, and is not related to any existing job.

Each table has as a common constraint a primary key as an ID column. An automatically generated trigger adds an incremented integer before any insert statement. The quantity column of the inventory table has the integer ‘1’ as a default—that is, if no quantity is added, it is assumed that a single unit was purchased. The foreign keys were created automatically from the relations: the job table has the foreign key ‘customer\_id’; the subcontract table has foreign keys ‘sub\_id’ and ‘job\_id’; and the expense table has foreign keys ‘job\_id’, ‘customer\_id’, and ‘inventory\_id’. Beyond the primary keys of each table, not null constraints were applied to the name and descriptive fields of all related tables. The cost field from expense and the email field from customer were also made mandatory.

# Diagram Description automatically generatedPhysical Entity Relationship Diagram

# Create Table Statements

-- Generated by Oracle SQL Developer Data Modeler 21.2.0.183.1957

-- at: 2021-12-07 08:39:01 EET

-- site: Oracle Database 11g

-- type: Oracle Database 11g

CREATE TABLE customer (

id INTEGER NOT NULL,

lastname VARCHAR2(20 CHAR) NOT NULL,

firstname VARCHAR2(20 CHAR) NOT NULL,

address VARCHAR2(50 CHAR),

telephone INTEGER,

email VARCHAR2(40 CHAR) NOT NULL,

priority VARCHAR2(10 CHAR)

);

COMMENT ON COLUMN customer.priority IS

'low/high/very low/very high/(null)';

ALTER TABLE customer ADD CONSTRAINT customer\_pk PRIMARY KEY ( id );

CREATE TABLE expense (

id INTEGER NOT NULL,

description VARCHAR2(30 CHAR) NOT NULL,

cost NUMBER(10, 2) NOT NULL,

job\_id INTEGER,

customer\_id INTEGER,

inventory\_id INTEGER,

reimbursable VARCHAR2(8)

);

COMMENT ON COLUMN expense.description IS

'job type (i.e. router box installation)';

COMMENT ON COLUMN expense.reimbursable IS

'(yes/no/null)';

ALTER TABLE expense ADD CONSTRAINT expense\_pk PRIMARY KEY ( id );

CREATE TABLE inventory (

id INTEGER NOT NULL,

item VARCHAR2(20 CHAR) NOT NULL,

quantity INTEGER,

consumable VARCHAR2(8 CHAR)

);

COMMENT ON COLUMN inventory.item IS

'inventory item description ';

COMMENT ON COLUMN inventory.consumable IS

'equipment or depletable resource (yes/no)';

ALTER TABLE inventory ADD CONSTRAINT inventory\_pk PRIMARY KEY ( id );

CREATE TABLE job (

id INTEGER NOT NULL,

description VARCHAR2(50 CHAR) NOT NULL,

job\_date DATE,

address VARCHAR2(50 CHAR) NOT NULL,

distance\_km NUMBER(10, 2),

drive\_time\_hrs NUMBER(10, 2),

start\_time TIMESTAMP,

end\_time TIMESTAMP,

status VARCHAR2(10 CHAR),

customer\_id INTEGER NOT NULL,

payment NUMBER(10, 2)

);

COMMENT ON COLUMN job.description IS

'job type (i.e. router box installation)';

COMMENT ON COLUMN job.job\_date IS

'starting date of job';

COMMENT ON COLUMN job.distance\_km IS

'distance (km) from office to job address.';

COMMENT ON COLUMN job.drive\_time\_hrs IS

'estimated time it will take to get to job address.';

COMMENT ON COLUMN job.start\_time IS

'starting time of job';

COMMENT ON COLUMN job.end\_time IS

'ending time of job (initially estimate)';

COMMENT ON COLUMN job.status IS

'completed/active/delayed (or null)';

ALTER TABLE job ADD CONSTRAINT job\_pk PRIMARY KEY ( id );

CREATE TABLE pricelist (

id INTEGER NOT NULL,

work\_type VARCHAR2(15 CHAR),

rate\_per\_hr NUMBER(10, 2)

);

COMMENT ON COLUMN pricelist.work\_type IS

'what type of work the subcontractor is best at';

COMMENT ON COLUMN pricelist.rate\_per\_hr IS

'includes approximate estimate of material cost for typical job of given time allotment';

ALTER TABLE pricelist ADD CONSTRAINT pricelist\_pk PRIMARY KEY ( id );

CREATE TABLE subcontractor (

id INTEGER NOT NULL,

specialty VARCHAR2(15 CHAR),

sub\_name VARCHAR2(20 CHAR) NOT NULL,

sub\_first VARCHAR2(20 CHAR) NOT NULL,

rate\_per\_hr NUMBER(10, 2),

wks\_until\_able INTEGER,

sub\_address VARCHAR2(50 CHAR),

sub\_tel INTEGER,

sub\_email VARCHAR2(40 CHAR)

);

COMMENT ON COLUMN subcontractor.specialty IS

'what type of work the subcontractor is best at';

ALTER TABLE subcontractor ADD CONSTRAINT subcontractor\_pk PRIMARY KEY ( id );

CREATE TABLE subcontract (

sub\_id INTEGER NOT NULL,

job\_id INTEGER NOT NULL

);

ALTER TABLE subcontract ADD CONSTRAINT job\_subcontractors\_pk PRIMARY KEY ( sub\_id,

job\_id );

ALTER TABLE expense

ADD CONSTRAINT expense\_customer\_fk FOREIGN KEY ( customer\_id )

REFERENCES customer ( id );

ALTER TABLE expense

ADD CONSTRAINT expense\_inventory\_fk FOREIGN KEY ( inventory\_id )

REFERENCES inventory ( id );

ALTER TABLE expense

ADD CONSTRAINT expense\_job\_fk FOREIGN KEY ( job\_id )

REFERENCES job ( id );

ALTER TABLE job

ADD CONSTRAINT job\_customer\_fk FOREIGN KEY ( customer\_id )

REFERENCES customer ( id );

ALTER TABLE subcontract

ADD CONSTRAINT job\_sub\_job\_fk FOREIGN KEY ( job\_id )

REFERENCES job ( id );

ALTER TABLE subcontract

ADD CONSTRAINT job\_sub\_sub\_fk FOREIGN KEY ( sub\_id )

REFERENCES subcontractor ( id );

# Source Code for Triggers

CREATE SEQUENCE customer\_id\_seq START WITH 1 NOCACHE ORDER;

CREATE OR REPLACE TRIGGER customer\_id\_trg BEFORE

INSERT ON customer

FOR EACH ROW

WHEN ( new.id IS NULL )

BEGIN

:new.id := customer\_id\_seq.nextval;

END;

/

CREATE SEQUENCE expense\_id\_seq START WITH 1 NOCACHE ORDER;

CREATE OR REPLACE TRIGGER expense\_id\_trg BEFORE

INSERT ON expense

FOR EACH ROW

WHEN ( new.id IS NULL )

BEGIN

:new.id := expense\_id\_seq.nextval;

END;

/

CREATE SEQUENCE inventory\_id\_seq START WITH 1 NOCACHE ORDER;

CREATE OR REPLACE TRIGGER inventory\_id\_trg BEFORE

INSERT ON inventory

FOR EACH ROW

WHEN ( new.id IS NULL )

BEGIN

:new.id := inventory\_id\_seq.nextval;

END;

/

CREATE SEQUENCE job\_id\_seq START WITH 1 NOCACHE ORDER;

CREATE OR REPLACE TRIGGER job\_id\_trg BEFORE

INSERT ON job

FOR EACH ROW

WHEN ( new.id IS NULL )

BEGIN

:new.id := job\_id\_seq.nextval;

END;

/

CREATE SEQUENCE pricelist\_id\_seq START WITH 1 NOCACHE ORDER;

CREATE OR REPLACE TRIGGER pricelist\_id\_trg BEFORE

INSERT ON pricelist

FOR EACH ROW

WHEN ( new.id IS NULL )

BEGIN

:new.id := pricelist\_id\_seq.nextval;

END;

/

CREATE SEQUENCE subcontractor\_id\_seq START WITH 1 NOCACHE ORDER;

CREATE OR REPLACE TRIGGER subcontractor\_id\_trg BEFORE

INSERT ON subcontractor

FOR EACH ROW

WHEN ( new.id IS NULL )

BEGIN

:new.id := subcontractor\_id\_seq.nextval;

END;

# Insert and Update Statements

insert into customer (lastname, firstname, address, telephone, email, priority)

values ('Giannopoulos', 'Steven', '6987 Fairview, Chicago, IL', 93828170382,

'steve.gian@hotmail.com', 'Low');

update customer

set telephone = 9382810382

where firstname = 'Steven';

update customer

set email = 'comacris01@gmail.com'

where firstname = 'Constantine';

insert into customer (lastname, firstname, address, telephone, email, priority)

values ('Moschovis', 'Sotiria', '6127 N Leader Ave, Chicago, IL', 7739128761,

'moschovis@gmail.com', 'High');

insert into customer (lastname, firstname, address, telephone, email, priority)

values ('Macris', 'Constantine', '6127 N Leader Ave, Chicago, IL', 6973187491,

'macris.constantine@gmail.com', 'High');

insert into customer (lastname, firstname, address, telephone, email)

values ('Clarkson', 'Brent', '9 Westpark Dr, Park Ridge, IL', 0986765453,

'blarckson@gmail.com');

insert into customer (lastname, firstname, address, telephone, email, priority)

values ('Smith', 'Mark', '8 Grove Ave, Cedarville, OH', 7739128761,

'smarkson@yahoo.com', 'Very Low');

insert into customer (lastname, firstname, address, telephone, email, priority)

values ('Romisch', 'Tim', '6 Westwood Dr, Bloomfield Hills, MI', 7739128761,

'moschovis@gmail.com', 'High');

update customer

set email = 'timnpeggy@gmail.com'

where lastname = 'Romisch';

delete from customer

where firstname = 'Constantine';

insert into expense (description, cost, reimbursable)

values ('gas', 30, 'yes');

insert into expense (description, cost, customer\_id, reimbursable)

values ('modem', 80, 31, 'yes');

insert into expense (description, cost, job\_id, reimbursable)

values ('3m ethernet cables', 12.75, 7, 'yes');

insert into inventory (item, quantity, consumable)

values ('pack of screws', 4, 'yes');

insert into inventory (item, quantity, consumable)

values ('lg moving boxes', 3, 'yes');

insert into inventory (item, quantity, consumable)

values ('box of zipties', 2, 'yes');

insert into inventory (item, quantity, consumable)

values ('labelling tape', 6, 'yes');

insert into inventory (item, quantity, consumable)

values ('complete driver set', 1, 'no');

insert into expense (description, cost, inventory\_id, reimbursable)

values ('screws', 10.00, 4, 'yes');

insert into expense (description, cost, job\_id, inventory\_id, reimbursable)

values ('labelling tape', 8.00, 7, 15, 'yes');

insert into job (description, address, distance\_km, drive\_time\_hrs,

start\_time, end\_time, status, customer\_id, payment)

values ('Router box re-location', '205 W Wacker Dr Suite 100, Chicago, IL', 24,

2, '02-MAR-2020 08:01', '01-MAY-2020 08:23', 'COMPLETE', 34, 1200);

insert into job (description, address, distance\_km, drive\_time\_hrs,

start\_time, end\_time, status, customer\_id, payment)

values ('router box installation', '815 N Slacker Ln, Chicago, IL', 78,

5.5, '18-FEB-2020 10:00', '19-FEB-2020 03:00', 'COMPLETE', 34, 300);

insert into job (description, address, distance\_km, drive\_time\_hrs,

start\_time, status, customer\_id)

values ('LAN system configuration', '998 N Seeker Bvd, Niles, IL', 12,

1.2, '06-MAY-2020 12:01', 'UPCOMING', 31);

insert into job (description, address, distance\_km, drive\_time\_hrs,

start\_time, end\_time, status, customer\_id, payment)

values ('telecom installation', '8 Grove Ave, Cedarville, OH', 350,

6.5, '11-JUN-2020 05:00', '13-JUN-2020 08:00', 'COMPLETE', 35, 2200);

insert into job (description, address, distance\_km, drive\_time\_hrs,

start\_time, end\_time, status, customer\_id, payment)

values ('network wiring', '6987 Fairview, Chicago, IL', 21,

0.5, '20-JUN-2020 05:00', '21-JUN-2020 08:00', 'COMPLETE', 31, 650);

insert into job (description, address, distance\_km, drive\_time\_hrs,

start\_time, end\_time, status, customer\_id, payment)

values ('network system repair', '6 Westwood Dr, Bloomfield Hills, MI', 378,

4.5, '13-OCT-2021 09:00', '14-OCT-2021 08:00', 'COMPLETE', 36, 500);

delete from job

where description like '%LAN%' and customer\_id = 31;

update job

set end\_time = to\_timestamp('13-OCT-2021 16:45', 'DD-MON-YYYY HH24:MI')

where id = 21;

insert into pricelist (work\_type, rate\_per\_hr)

values ('computer system installation', 50);

insert into pricelist (work\_type, rate\_per\_hr)

values ('applicance re-location', 30);

insert into pricelist (work\_type, rate\_per\_hr)

values ('moderately useful assistance', 22);

insert into pricelist (work\_type, rate\_per\_hr)

values ('software debugging', 35);

insert into pricelist (work\_type, rate\_per\_hr)

values ('telecom system maintenance', 40);

insert into subcontractor (specialty, sub\_name, sub\_first, rate\_per\_hr,

wks\_until\_able, sub\_address, sub\_tel, sub\_email)

values ('electrical wiring', 'Hanson', 'James', 35, 2,

'12 Clearview Dr 743, Wheaton, IL', 7739183891, 'hansjam@gmail.com');

insert into subcontractor (specialty, sub\_name, sub\_first, rate\_per\_hr,

wks\_until\_able, sub\_address, sub\_tel, sub\_email)

values ('being available', 'Macris', 'Constantine', 20, 0,

'Ippolitou 10, 11255 Athens, Greece', 7739183891, 'comacris01@gmail.com');

insert into subcontractor (specialty, sub\_name, sub\_first, rate\_per\_hr,

wks\_until\_able, sub\_address, sub\_tel, sub\_email)

values ('digital media software', 'George', 'Ryan', 33, 5,

'9124 Bognatti Ln, San Diego, FL', 8171989023, 'hihellothere@gmail.com');

insert into subcontractor (specialty, sub\_name, sub\_first, rate\_per\_hr,

wks\_until\_able, sub\_address, sub\_tel, sub\_email)

values ('heavy moving', 'Paul', 'Chris', 45, 10,

'4931 W Northshire Ave, Cleveland OH', 7739183891, 'pgeorgeous@acg.edu');

insert into subcontractor (specialty, sub\_name, sub\_first, rate\_per\_hr,

wks\_until\_able, sub\_address, sub\_tel, sub\_email)

values ('audiovisual cabling', 'Gaye', 'Marvin', 99, 20,

'605 Stoneybrook Rd, W Hollywood, CA', 7739183891, 'comacris01@gmail.com');

update subcontractor

set sub\_tel = 9912380981

where sub\_name = 'Paul';

alter table subcontractor

modify specialty varchar(30);

insert into subcontract (sub\_id, job\_id)

values (3,7);

insert into subcontract (sub\_id, job\_id)

values (3,21);

insert into subcontract (sub\_id, job\_id)

values (1,20);

insert into subcontract (sub\_id, job\_id)

values (8,7);

insert into subcontract (sub\_id, job\_id)

values (9,19);

update job

set description = 'router box re-location'

where description = 'Router box re-location';

alter table pricelist

add constraint not\_negative check (rate\_per\_hr >=0);

alter table subcontractor

add constraint positive\_num check (rate\_per\_hr >=0);

alter table job

add constraint non\_negative check (payment >= 0);

alter table inventory

add constraint greater\_zero check (quantity >= 0);

alter table inventory

modify quantity default 1;

update job

set payment = -40;

# Select Statements and Results

**--displays jobs with a status of 'complete' and payment over 1000**

SELECT description, status, payment

FROM job

WHERE status = 'COMPLETE'

AND payment > 1000;

Table

Description automatically generated

**--joins and displays customers with corresponding jobs**

SELECT firstname || ' ' || lastname as "CUSTOMER NAME",

description as "JOB DESCRIPTION", status, payment

FROM customer

JOIN job on (job.customer\_id = customer.id);

Table

Description automatically generated

**--repeats previous statement but includes customer with no job assignment**

SELECT firstname || ' ' || lastname as "CUSTOMER NAME",

description as "JOB DESCRIPTION", status, payment

FROM customer

LEFT OUTER JOIN job on (job.customer\_id = customer.id);

Table

Description automatically generated

**--displays the sum of job payments from customers, grouped by customer**

SELECT customer.id, firstname || ' ' || lastname AS "NAME",

Count(payment) AS "NO. OF PAYMENTS",

SUM(payment) AS "TOTAL PAYMENT"

FROM job JOIN customer ON (job.customer\_id = customer.id)

GROUP BY customer.id, firstname, lastname;

Table

Description automatically generated

**--enhances previous statement to include group condition**

SELECT customer.id, firstname || ' ' || lastname AS "NAME",

Count(payment) AS "NO. OF PAYMENTS",

SUM(payment) AS "TOTAL PAYMENT"

FROM job JOIN customer ON (job.customer\_id = customer.id)

GROUP BY customer.id, firstname, lastname

HAVING Sum(payment) > 2000;

Table, calendar

Description automatically generated

**--displays duration (or planned duration) of active and completed jobs**

SELECT description, CONCAT(to\_char(start\_time, 'DD-MON-YYYY HH24:MI --- '),

to\_char(end\_time, 'DD-MON-YYYY HH24:MI'))

AS "JOB DURATIONS"

FROM job

WHERE status != 'UPCOMING'

ORDER BY start\_time ASC;

Table

Description automatically generated

**--displays subcontractors joined with their respective jobs and using a subquery from the expenses**

**--table to specify only the subcontracted jobs that incurred additional expenses over 10 dollars**

SELECT sub\_name, specialty, job\_id, description

FROM subcontractor

JOIN subcontract on (subcontract.sub\_id = subcontractor.id)

JOIN job on (subcontract.job\_id = job.id)

WHERE job\_id IN (SELECT job\_id FROM expense WHERE cost > 10);

Table, calendar

Description automatically generated

# Applications Screenshots and Descriptions

Below is an image of the menu from the java application:

Text

Description automatically generated

Results from the first selection:

Text

Description automatically generated

Results from the second selection:

Text

Description automatically generated

Results from the third selection:

Text

Description automatically generated

Results from the fourth selection:

Text

Description automatically generated

Results from the fifth selection:

Text

Description automatically generated

Results from the sixth selection:

Text

Description automatically generated

Results from the seventh selection:

Text

Description automatically generated

Results from choosing “8.” from the menu:

Text

Description automatically generated

Result of input not matching menu selection:

Text

Description automatically generated

# Source Code for the Java Application

package com.company;

import java.sql.\*;

import java.util.Scanner;

public class Main {

static Connection connection;

public static void main(String[] args) {

try {

Class.forName("oracle.jdbc.driver.OracleDriver");

System.out.println("Oracle driver loaded");

} catch (ClassNotFoundException e) {

System.out.println("Oracle Driver not found");

System.exit(0);

}

try {

connection =

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1522/xepdb1",

"field\_service", "oracle");

System.out.println("Connection to DB user field\_service, successful!");

} catch (SQLException e) {

System.out.println("Unable to establish the connection..." + e.getMessage());

System.exit(0);

}

displayMenu();

}

public static void displayMenu() {

Scanner input = new Scanner(System.in);

int option;

System.out.println("------<>--------------<>------");

System.out.println("Triturion Database Information");

System.out.println("------<>--------------<>------");

System.out.println("1. Completed jobs that paid more than 1000");

System.out.println("2. Customer names linked with their jobs");

System.out.println("3. Customers and include those not linked to a job");

System.out.println("4. Count and sum of payments grouped by customer");

System.out.println("5. Count and sum of payments > 2000, grouped by customer ");

System.out.println("6. Duration (or estimate) of completed and active jobs");

System.out.println("7. Subcontractors and their jobs with expenses > 10");

System.out.println("8. EXIT");

System.out.println("<>------------------<>------------------<>");

System.out.println("Select a menu choice to display the query: ");

option = input.nextInt();

switch(option) {

case 1: executeView1(connection); break;

case 2: executeView2(connection); break;

case 3: executeView3(connection); break;

case 4: executeView4(connection); break;

case 5: executeView5(connection); break;

case 6: executeView6(connection); break;

case 7: executeView7(connection); break;

case 8: System.exit(0); break;

default: System.out.println("Invalid Option");

}

}

public static void executeView1(Connection connection) {

try {

Statement stmt;

ResultSet rs;

stmt = connection.createStatement();

rs = stmt.executeQuery("select \* from view1");

System.out.println("\n");

while (rs.next()) {

System.out.println(rs.getString("DESCRIPTION") +

" " + rs.getString("STATUS") + " "

+ rs.getDouble("PAYMENT"));

}

} catch (SQLException e) {

System.out.println("Something went wrong with Query 1: " + e.getMessage());

}

}

public static void executeView2(Connection connection) {

try {

Statement stmt;

ResultSet rs;

stmt = connection.createStatement();

rs = stmt.executeQuery("select \* from view2");

System.out.println("\n");

while (rs.next()) {

System.out.println(rs.getString("Customer Name") + " " +

rs.getString("JOB DESCRIPTION") +

" " + rs.getString("STATUS") + " "

+ rs.getDouble("PAYMENT"));

}

} catch (SQLException e) {

System.out.println("Something went wrong with Query 2: " + e.getMessage());

}

}

public static void executeView3(Connection connection) {

try {

Statement stmt;

ResultSet rs;

stmt = connection.createStatement();

rs = stmt.executeQuery("select \* from view3");

System.out.println("\n");

while (rs.next()) {

System.out.println(rs.getString("Customer Name") + " " +

rs.getString("Job DESCRIPTION") +

" " + rs.getString("STATUS") + " "

+ rs.getDouble("PAYMENT"));

}

} catch (SQLException e) {

System.out.println("Something went wrong with Query 3: " + e.getMessage());

}

}

public static void executeView4(Connection connection) {

try {

Statement stmt;

ResultSet rs;

stmt = connection.createStatement();

rs = stmt.executeQuery("select \* from view4");

System.out.println("\n");

while (rs.next()) {

System.out.println(rs.getInt("ID") + " " +

rs.getString("NAME") + " " +

rs.getInt("NO. OF PAYMENTS") +

" " + rs.getDouble("TOTAL PAYMENT"));

}

} catch (SQLException e) {

System.out.println("Something went wrong with Query 4: " + e.getMessage());

}

}

public static void executeView5(Connection connection) {

try {

Statement stmt;

ResultSet rs;

stmt = connection.createStatement();

rs = stmt.executeQuery("select \* from view5");

System.out.println("\n");

while (rs.next()) {

System.out.println(rs.getInt("ID") + " " +

rs.getString("NAME") + " " +

rs.getInt("NO. OF PAYMENTS") +

" " + rs.getDouble("TOTAL PAYMENT"));

}

} catch (SQLException e) {

System.out.println("Something went wrong with Query 5: " + e.getMessage());

}

}

public static void executeView6(Connection connection) {

try {

Statement stmt;

ResultSet rs;

stmt = connection.createStatement();

rs = stmt.executeQuery("select \* from view6");

System.out.println("\n");

while (rs.next()) {

System.out.println(rs.getString("DESCRIPTION") + " "

+ rs.getString("JOB DURATIONS"));

}

} catch (SQLException e) {

System.out.println("Something went wrong with Query 6: " + e.getMessage());

}

}

public static void executeView7(Connection connection) {

try {

Statement stmt;

ResultSet rs;

stmt = connection.createStatement();

rs = stmt.executeQuery("select \* from view7");

System.out.println("\n");

while (rs.next()) {

System.out.println(rs.getString("SUB\_NAME") + " " +

rs.getString("SPECIALTY") + " " +

rs.getInt("JOB\_ID") +

" " + rs.getString("DESCRIPTION"));

}

} catch (SQLException e) {

System.out.println("Something went wrong with Query 7: " + e.getMessage());

}

}

}

Thank you for teaching us this so patiently this semester:)