**Week 5 Homework**

The following practice may need to be completed using the databases hosted on wmc3317-2 using WB. Use WB whether you are running queries or creating a relational schema (a data model, essentially). Submit homework as a SQL script file. To mark the answers I will copy+paste the SQL script into my WB and execute the code. You could even write essay answers in an MS Word file or the SQL script itself – or in a separate MS Word/PDF file. If there is any dirty data, clean it by making reasonable assumptions.

1. **Consider the Sakila data model used in class.** (you cannot test these queries since either the schema does not match or there is no data)

Customer(custid (PK), firstname)

Rental (rentalid (PK), invid (FK), customerid (FK), dtMonth, dtYear, dtDay)

Inventory (invid (PK), filmid (FK))

Film (filmid (PK), length, name, cost, releaseMth, releaseYr)

Film\_actor(filmid (PK FK), actorid (PK FK))

Actor (actorid (PK), actorname)

Note: Film.Cost refers to the film studio’s cost of making the film

1. We need to analyze whether the cost of making a movie does indeed predict the rental volume (number of rentals) in the first three months after its release month, create a query that lists each movie, its name, alongside its cost, and the rental volume in the three months after its release. *For simplicity, assume no film is released from Oct to December of any year.*

select name, cost, count(distinct (filmid))

from rental as r, film as f, inventory as i

where  r.invid=i.invid

and   i.filmid=f.filmid

and   r.dtMonth > f.releaseMth

and  (r.dtMonth - f.releaseMth) <=3

            and   r.dtYear=f.releaseYr;

1. Now consider additional data acquired by mining tweets (from Twitter). numMentions is the number of times the actor has been mentioned in all the tweets for each month of each year. Make any other assumptions you need to.

ActorTweet(actorid (PK FK), numMentions, monthyear (PK FK))

Create the query that lists the actor, the total volume of rentals (number) during a month for all the movies that actor has acted in, the number of mentions for that actor on Twitter the previous month, and the total cost of making all those films he/she has acted in. *Assume no film is released in January.*

select

actor.actorid,

actor.actorname,

count(rentalid), #This month

ActorTweet.numMentions, #Previous Month

sum(Film.cost)

from Actor

inner join ActorTweet

on Actor.actorid = ActorTweet.actorid

inner join Film\_actor

on actor.actorid = Film\_actor.actorid

inner join Film

on Film\_actor.filmid = Film.filmid

inner join Inventory

on Film.filmid = Inventory.filmid

inner join Rental

on Inventory.invid = Rental.invid

where max(ActorTweet.monthyear) - 1

group by actor.actorid, rental.dtMonth, ActorTweet.monthyear

having max(rental.dtYear,rental.dtMonth);

# Assuming ActorTweet.monthyear is a date

1. Consider the following data model for the Employees database.

emp(empid (PK), name, bach, yrbach, mstr, yrmstr, phd, yrphd)

dept(deptid (PK), depname, empMgrid, dtstart, dtend)

deptEmp(deptid (PK, FK), empid (PK, FK), startDt (PK), endDt)

salary(empid (PK FK), salary, fromdt (PK), todt)

The columns - bach, mstr and phd - refer to the number of such degrees. The years (e.g., yrbach) pertain to the most recent ones (most recent bachelors, masters and Ph.D.). Write a query to list the employees, the number of degrees each has, the number of years since the most recent degree (use curryear()) to refer to the current year, and the most recent salary (use max(todt) to filter out the most recent salary for each employee).

select

empid, name,

(bach + mstr + phd) as NumofDegrees,

least(curryear() – yrbach, curryear() – yrmstr, curryear() – yrphd) as NumofYears,

#find the minimum number of years from current year to each degree of each employee

s1.salary

from emp e, salary s1, salary s2

where e.empid = s1.empid

and s1.empid = s2.empid

and s1.todt = (select max(todt) from salary s2)

group by e.empid

order by e.empid asc;

1. Create a data model for the Volunteer Club running case. Write the following queries based on that model.

**Data Model for Volunteer Club**

* **club\_member**

member\_number(PK),first\_name, last\_name, community\_name, street\_address, city, state, zip\_code, phone\_number, fax\_number, email\_address, news\_letter\_delivery\_method, member\_type, committee\_code(FK)

* **commitee**

committee\_code(PK), commitee\_name, mission, member\_number(FK)

* **committeevent**

committee\_code(PK FK), event \_code (PK FK)

* **donation\_1**

donation\_code(PK), amount, description, received, member\_number(FK), event \_code (FK), donor\_code(FK)

* **event**

event \_code (PK), event\_name, location, begin\_date\_time, end\_date\_time, description, committee\_code

* **memberevent**

member\_number (PK FK), event \_code (PK FK)

* **donation\_2**

donation\_code(PK), amount, description, received, event \_code (FK), charity\_code(FK)

* **charity**

charity\_code(PK), name, street\_address, city, state, zip\_code, phone\_number, first\_name, last\_name, work\_desciption

* **donor**

donor\_code(PK), organization, address, phone\_nuumber, contact\_last\_name, contact\_first\_name, contact\_home\_phone\_no, contact\_cell\_no, contact\_work\_phone\_no, contact\_email

1. List all events, their location, and the total value of the donations received for each event, in descending order of the total value received?

#Assuming that the received variable is a Boolean variable, recording whether the donation has been received (yes or null)

#Assuming vc is the database

use vc;

select

event.event\_code,

event.event\_name,

event.location,

donation\_1.received,

donation\_1.amount

from event

inner join donation\_1

on event.event\_code = donation\_1.event\_code

where received = "yes"

group by event\_code

order by amount desc;

1. How many donors participated in each event?

#Assuming vc is the database name

use vc;

select

event.event\_code,

event.event\_name,

count(donation\_1.donor\_code) as NumofDonors

from event

inner join donation\_1

on event.event\_code = donation\_1.event\_code

group by event\_code;

1. Which events received no donations?

#Assuming that the received variable is a Boolean variable, recording whether the donation has been received (yes or null)

#Assuming vc is the database name

#Assuming "no donation" means that the event raised no money and no donation was received for that event

use vc;

select

event.event\_code,

event.event\_name,

donation\_1.amount,

donation\_1.received

from event

inner join donation\_1

on event.event\_code = donation\_1.event\_code

where donation\_1.amount = 0 and donation\_1.received = null

group by event.event\_code

order by donation\_1.amount desc;

1. Which club members participated in more than three events?

#Assuming vc is the database name

use vc;

select

club\_member.member\_number,

club\_member.first\_name,

club\_member.last\_name,

count(donation\_1.event\_code) as NumofEvents

from club\_member

inner join donation\_1

on club\_member.member\_number = donation\_1.member\_number

group by club\_member.member\_number

having NumofEvents > 3

order by NumofEvents desc;

1. List each event that has not taken place yet and the names and phone numbers of each member who is working on it?

#Assuming vc is the database name

use vc;

select

event.event\_code,

event.event\_name,

event.begin\_date\_time,

event.end\_date\_time,

club\_member.member\_number,

club\_member.first\_name,

club\_member.last\_name,

club\_member.phone\_number

from event

inner join committeevent

on event.event\_code = committeevent.event\_code

inner join committee

on committeevent.committee\_code = committee.committee\_code

inner join club\_member

on committee.member\_number = club\_member.member\_number

where (event.begin\_date\_time >= curdate() and event.begin\_date\_time >= curtime()) or event.begin\_date\_time > curdate();