**Week 2 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.

1. **Database: ‘bigpvfc’. Copy/paste your answers to the queries below after testing that they work.**
2. List each customer name, address and the number of orders each has placed. (use tables: customer\_t, order\_t)

Use bigpvfc;

Select customer\_t.CustomerName, customer\_t.CustomerAddress, Count(OrderID) as NumberOrdered

From order\_t

Inner join customer\_t

on Order\_t.CustomerID=Customer\_t.CustomerID

Group By CustomerName;

1. List only name and address of customers who have placed more than two orders.

Use bigpvfc;

Select customer\_t.CustomerName, customer\_t.CustomerAddress, Count(OrderID) as NumberOrdered

From order\_t

Inner join customer\_t

on Order\_t.CustomerID=Customer\_t.CustomerID

Group By CustomerName

Having count(OrderID) >2;

1. List the running total of the number of total number of orders placed till date as they come in (use table: order\_t).

Use bigpvfc;

set @runningtotal :=0;

Select ordernew.OrderDate,

(@runningtotal := @runningtotal + ordernew.count) as Running\_Total

From

(select count(OrderID) as count, OrderDate

from order\_t group by OrderDate) as ordernew

order by OrderDate;

1. **Database: ‘sakila’. Copy/paste your answers to the queries below after testing that they work.**

While an employee at Sakila, you have been asked to integrate the ‘film’ table with an additional database of movies from <http://www.hometheaterinfo.com/dvdlist.htm>. The reason is that Sakila may expect a large shipment of additional movies and may add it to their catalog. Import the entire film file from the above URL (.csv format), store it as a separate table in your database. Then select a set of movies from the imported list based on an appropriate criteria and update the ‘film’ table with the additional movies. This is an open-ended question. The entire procedure should be saved as an SQL script – so that you can always re-use it in the future.

This is messy. Do what you can. Write what you did and what you found. And make reasonable assumptions.

1. Download and save the .txt file of the dvdlist
2. Create a table on my personal database named new\_film to import the data from the dvdlist .txt file

Use db\_vphan;

Drop table new\_film;

Create table new\_film (

DVD\_Title varchar (255),

Studio varchar (50),

Released varchar (10),

Status varchar (50),

Sound decimal (2,2),

Versions varchar (5),

Price decimal (6,2),

Rating varchar (50),

Year varchar (10),

Genre varchar (20),

Aspect varchar (20),

UPC varchar (20),

DVD\_ReleaseDate timestamp,

ID decimal (10) NOT NULL,

Timestamp timestamp,

Primary Key (ID));

1. Load the data from dvdlist .txt file to the new table

LOAD DATA LOCAL INFILE

'C:/Users/Vivian/Desktop/BUS 464/Week 2/dvd\_csv.txt' INTO TABLE new\_film

FIELDS TERMINATED BY ','

ENCLOSED BY '\''

LINES TERMINATED BY '\n'

IGNORE 1 LINES;

#We notice that there are 273,781 records in the dvdlist. However, we cannot load all of those records to the new table because there are many data entry errors. The data needs to be cleaned up and formatted so it can be properly imported to MySQL. We set ID as our Primary Key because it appears that each movie in the list has a unique ID number.

1. Create a copy of the film table from sakila database onto my personal database

Use db\_vphan;

Create table film as (select \* from sakila.film);

1. We find the appropriate criteria for updating the film table

#We notice from the table film that the highest replacement\_cost for a movie is no more than $29.99. We confirm this by running:

Use db\_vphan;

Select distinct replacement\_cost from film order by replacement\_cost desc;

#From this observation, we assume that the company does not want movies with very high replacement costs (i.e. more than $29.99). Thus we set the replacement\_cost as our criteria.

1. We update the film table with a set of movies that has the price <= $29.99

Update film

Inner Join new\_film

on film.replacement\_cost = new\_film.Price

Set film.title = new\_film.DVD\_Title, film.release\_year = new\_film.Year, film.replacement\_cost = new\_film.Price, film.rating = new\_film.Rating

where new\_film.Price <= 29.99;