University of Waterloo
Midterm Examination
Winter Term 2016

Student Name: ____

Student Signature: ____

Student Identification Number: ____

Course Section/Instructor (Please Circle): 001/Ward

Course Number:	E&CE 356
Course Title:	Database Systems
Sections:	001
Instructors:	Paul Ward
Date of Exam:	Thursday, February 25$^{\text{th}}$ 2016
Time Period:	5:35 AM to 6:50 AM
Duration of Exam:	75 minutes
Number of Exam Pages (including cover sheet):	8 pages / 3 questions
Exam Type:	Closed Book
Additional Materials Allowed:	None

General Notes

1. No electronic devices including no calculators.
2. Read the entire questions *carefully* before answering.
3. State any assumptions clearly and whenever needed.
4. No questions permitted. When in doubt, make an assumption, write it down.
5. Some SQL reminders are listed on Page 8.
6. *qui audet adipiscitur*

Question	**Maximum**	**Score**					**Total**
1	25	(a)	(b)	(c)	(d)	(e)	
2	25	(a)	(b)	(c)	(d)	(e)	
3	25						
Total	75						

1. [25 marks] SQL Comprehension: Consider the following relations:

Person				
ID int	FirstName char(20)	LastName char(20)	Sex char(1)	BirthDate date
123	Fred	Jones	M	1/10/1974
456	Mary	Jane	F	23/4/1977
789	Jimbo	Jones	M	16/3/1994
101	Alex	Jane	F	13/9/1996
112	Josh	Jones	M	3/2/2015

Parent	
ParentID int	ChildID int
123	789
456	101
123	101
101	112

(a) What is the following query computing (in one sentence) and what is its output for this dataset:

```
select count(distinct ChildID) from Parent;
```

(b) What is the following query computing (in one sentence), what is its output for this dataset, and what happens if "distinct" is omitted:

```
select count(distinct ParentID) as Total from Parent inner join Person on ID=ParentID
        where Sex='F';
```

(c) What is the following query computing (in one sentence), and what is its output for this dataset:

```
select count(distinct P1.ParentID) from Parent as P1 inner join Parent as P2
        where P1.ChildID=P2.ParentID;
```

(d) What is the following query computing (in a sentence) and what is its output for this dataset:

```
select P.FirstName, P.LastName, P3.FirstName from Person as P inner join Parent as P1
                                                            on P.ID=P1.ParentID
                                        inner join Parent as P2 on P1.ChildID=P2.ParentID
                                        inner join Person as P3 on P2.ChildID=P3.ID
                                        where P.LastName = P3.LastName;
```

(e) What are plausible primary and foreign keys for these two relations? Justify.

2. [25 marks] Query Creation:

Consider the following relational schema used to describe a car-rental database (primary keys are underlined):
Customer (c_num, c_name, city)
Car (licence, make, model, year)
Pickup (r_num, c_num, licence, fee, date, time, city)
Dropoff (r_num, date, time, city)
Explanation:

- Customer defines a unique customer number, his/her name, and location
- Car defines a unique car, together with information about the vehicle
- Pickup defines any rental contract in which the vehicle has been picked up, when and where, and for how much
- Dropoff contains the corresponding information for the return of a vehicle
- The attributes c_num and licence in Pickup are foreign keys that reference the primary keys of Customer and Car, respectively. Likewise, r_num in Dropoff is a foreign key that references the primary key in Pickup.

(a) Using SQL or relational algebra, find the names of customers who have rented a car at least five times in the last year.

(b) Using SQL or relational algebra, identify how many cars are currently out on a rental from Waterloo (*i.e.*, any car that has been picked up in Waterloo but has yet to be dropped off).

(c) Using SQL or relational algebra, identify how many cars are dropped off in the same city in which they are picked up and how many are dropped off in a different city.

(d) Using SQL or relational algebra, identify the models of cars, by make, that generate the highest revenue (*i.e.* the sum of their fees is the highest).

(e) Using SQL or relational algebra, identify the most popular car make and model.

3. [25 marks] Functional Decomposition and Normalization: Consider a database that contains information about people living in Canada. A person has a social insurance number (SIN), first name (FirstName), last name (LastName), zero or more middle names (MiddleName1, MiddleName2, ...), and a home address comprising a street number (StreetNumber), a street name (StreetName), a city (City), a province (Province), a postal code (PostCode), and (optionally) an apartment number (AptNumber). In unnormalized form, the Person relation is as follows:

- Person(SIN, FirstName, MiddleName1, MiddleName2, ..., LastName, AptNumber, StreetNumber, StreetName, City, Province, PostCode)

The functional dependencies are as follows:

- SIN $\rightarrow$ FirstName, MiddleName(s), LastName
- SIN $\rightarrow$ AptNumber, StreetNumber, StreetName, City, Province, PostCode
- StreetNumber, StreetName, City, Province $\rightarrow$ PostCode
- PostCode $\rightarrow$ City, Province

Given these functional dependencies, define a BCNF set of relations to capture the "Person" relation. Justify in precise, formal terms (*i.e.*, based on specific functional dependencies), why you have the decomposition that you have. Identify any functional dependencies that will not be enforced by your BCNF relations.

Some SQL Reminders

DDL:

```
create table
drop table [if exists <table>]
alter table <table> add <attribute> <domain>
alter table <table> drop <attribute>
create view <view> as <query expression>

Domain Types: char(n), varchar(n), int, smallint, numeric(p,d)
              real, double precision, float(n), date, time

Integrity     not null, default <V>, primary key (<a1, ..., an>),
Constraints:  unique(<a1, ..., an>),
              foreign key (<am, ..., an>) references (<am’, ..., an’>)

Referential   on update <...>, on delete <...>
Actions:      cascade, set null, set default, no action

Check Constraints: check (<predicate>)
```

DML:

```
insert into <table> values (...)
update <table> set <attribute = ...> where <predicate>
delete from <table> where <predicate>
select <a1,...,an> from <t1,...,tn> where <predicate>

Options on selection attributes: distinct, as

Options on selection tables: natural join, inner join, outer join
                             <join type> on <predicate> using <attributes>

Set Operations: union, interset, except (use ‘all’ to retain duplicates)

Aggregate Functions: avg, min, max, sum, count

Grouping: group by, having, order by

Options on predicate: =, !=, <, >, <=, >=, in, not in, exists, not exists,
                      is null, is not null, between, like <string pattern>
```