Family Name	....
First Name	....

SCIT

School of Computing and Information Technology

ISIT312
Big Data Management

This paper is for students studying at the Singapore Institute of Management Pte Ltd.

SAMPLE FINAL EXAMINATION PAPER

Exam value: **40% of the subject assessment**

Marks available: **40 marks.**

Question 1 (6 marks)

The following example depicts an outer join. Table X has two columns: A and B. Table Y has two columns: A and C. Table Z is an outer join of Table X and Table Y on column A. Use this example to explain how you implement an outer join in the MapReduce model.

You must specify the key-value data in the input and output of the Map and Reduce stages.

A	B
1	a
1	b
4	c

Table X

A	C
1	b
2	d
4	f

Table Y

A	B	C
1	a	b
1	b	b
2		d
4	c	f

Table Z

Question 2 (10 marks)

An objective of this task is to create a conceptual schema of a sample data warehouse domain described below. Read and analyse the following specification of a data warehouse domain.

A university would like to create a data warehouse to store information about the participation of the students in the lecture classes and later on to analyse the contents of a data warehouse.

It is expected that the planned data warehouse would contain historical information collected over a longer period of time.

A data warehouse supposed to contain information about the students enrolled in the courses offered by the university, lecture classes, lecture halls located in the university buildings and timetables of lecture classes.

The students attend the lecture classes in the lecture halls on the days and times determined by the timetables in the lecture halls.

The university uses a sophisticated electronic system that allows for automatic monitoring of the presence of each student in a lecture class, location of a student in a lecture hall (occupied seat number), and the activities performed by a student during a lecture class. The following activities can be detected: a student listens to a lecturer, a student is involved in a conversation with another student, or a student does not pay any attention to a lecture, for example a student can fall asleep for a while.

A student is described by a student number and full name. A student can be enrolled as either undergraduate or postgraduate student.

The lecture halls are located in the university buildings and described by a room number, building number, and the total number of seats. The buildings are described by a number, name, and the total number of levels. A lecture hall has a number of uniquely numbered seats.

The courses enrolled by the students are described by a course code, title and total number of credits points. The courses are included within the degrees. For simplicity reasons we assume that no course is shared by two or more degrees. The degrees are described by the titles and total fees.

An attendance of a student in a lecture class is described by the total time spent by a students, total time spent on listening to a lecturer, total time spent on conversations with the other students and total time when a student was not paying attention to a lecture class.

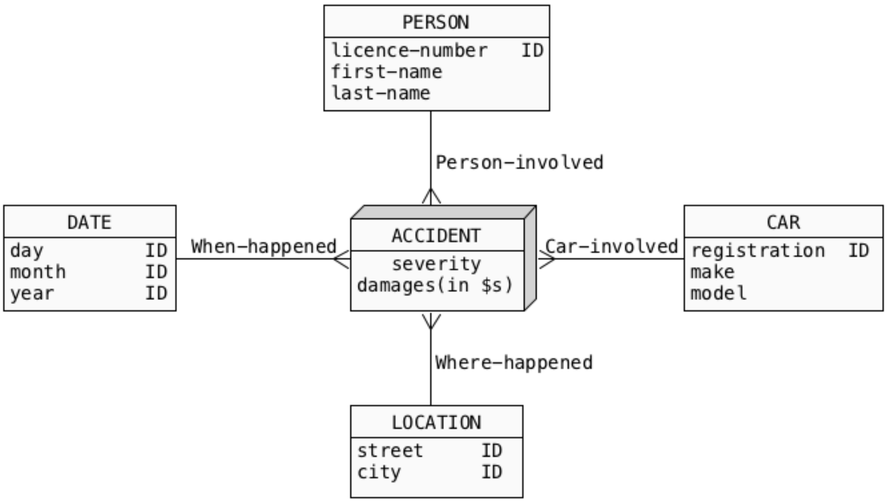
A data warehouse must be designed such it would be possible to easily implement the following classes of applications.

(1) *Find the total number attendances in the lecture classes per student, per course, per degree, per lecture hall, per building, per year, per session, per month, and per day. For example, it should be possible to find the total number of attendances in each month of 2017 in each course that belong to a give degree.*

(2) *Find the aggregations of the measures obtained from the student attendances in the lecture per student, per course, per degree, per lecture hall, per building, per year, per session, per month, and per day. For example, it should be possible to find the total time each student paid attention to a lecture class per subject, per session, per month in all lecture halls located in a building, or to find an average time a student did not pay attention to a lecture class per day, per month per subject.*

To draw a conceptual schema, use a graphical notation explained to you in a presentation 11 Conceptual Data Warehouse Design.

Question 3 (9 marks)
Consider the following four-dimensional data cube.



The data cube contains information about the car accidents that have happened in the past in the various locations and the people and cars involved in the accidents.

Assume that, the text files `date.tbl`, `location.tbl`, `person.tbl`, `car.tbl`, `accident.tbl` contain data obtained from the police descriptions of car accidents. The sample contents of the files `car.tbl` and `accident.tbl` are given below.

`car.tbl`

```
PKR856,Rolls Royce,Silver Shadow
UUQ076,Toyota,Corrolla
XYZ007,Gogo Mobile,Universal
...  ...  ...  ...  ...  ...
```

`accident.tbl`

```
25,SEP,2015,PKR856,Licence12345,Victoria St.,Sydney,serious,1256.67
13,MAR,2019,UUQ076,Licence6785,Bong Bong St.,Dapto,minor,100.00
01,FEB,2018,XYZ007,Licence9999,Liberation Ave.,average,894.50
...  ...  ...  ...  ...  ...
```

(1) Assume that the files `car.tbl` and `accident.tbl` are located at a folder `accidents` in a home folder of a user `jrg`. Write a piece of code that must be processed by a user `jrg` to load the files `car.tbl` and `accident.tbl` into HDFS. The files should be located in a folder `ACCIDENT` in HDFS. Location of a folder `ACCIDENT` in HDFS is up to you.

(2 marks)

(2) Write HQL statements to create the external tables that overlap the files `car.tbl` and `accident.tbl`. Use the following names as the names of external tables `ACCIDENT` and `CAR`. All other implementation details are up to you.

(4 marks)

(3) Write HQL statements to retrieve the following information from the tables `ACCIDENT` and `CAR`.

(i) Find the total number of accidents per car model, per year, per car model and year, and the total number of accidents.

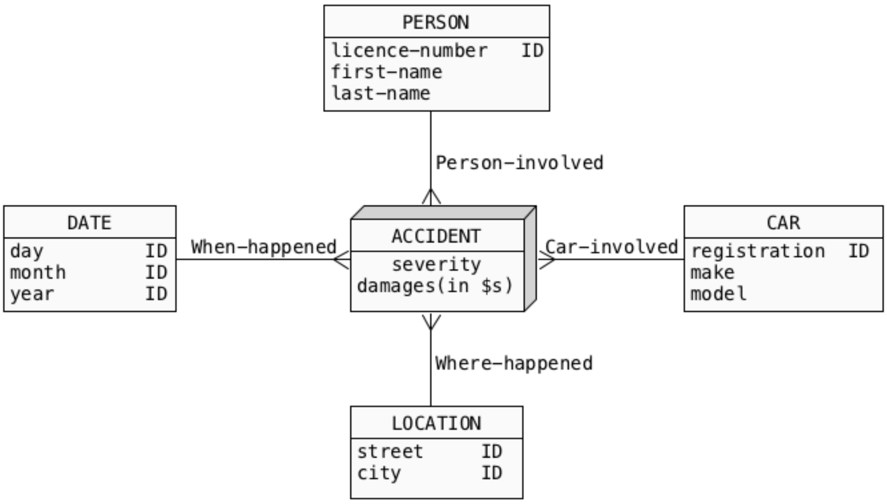
(ii) Find the average damages of all accidents that involved Toyota cars aggregated per year and per month, per year, and average damages of all accidents that involved Toyota cars.

(iii) Find the total number of serious accidents per year, per car make and model and per city.

(3 marks)

Question 4 (6 marks)

Implement as a single HBase table a database that contains information described by the following conceptual schema.



(1) Write HBase shell commands that create HBase table and load sample data into the table. Load into the table information about at least two accidents and two cars and one person involved both accidents. All other information is up to you.

(4 marks)

(2) Write HBase shell commands that implement the following queries.

(i) Find all information about the accidents having damages higher than 1000.

(ii) List the first and the last name of people involved in accidents in Sydney in 2019.

(2 marks)

Question 5 (5 marks)

Assume that the following files named students.txt, majors.txt and GPAs.txt, respectively, are in the HDFS directory hdfs://localhost:8080/records/.

```
--students.txt
--schema: studentid, name, majorid
1,Henry,100
2,Karen,100
3,Paul,101
4,Jimmy,102
5,Janice,102

--majors.txt
--schema: majorid, majorname
100,SoftwareEngineering
101,InformationManagement
102,BigData
103,CyberSecurity

-- GPAs.txt
--schema: studentid, marjorid, GPA
1,100,3.0
2,100,4.0
3,101,2.5
4,102,4.5
```

Write Pig-Latin statements that perform the following operations. For (5.2) and (5.3), also present the correct output.

(5.1) Load datasets by using the provided relation names and field names. The fields of each relation must have the suitable types.

(1 mark)

(5.2) Define a relation studentMajor which is the left outer join of students and majors. Then list the data in studentMajor.

(2 marks)

(5.3) Define a relation GPAbyMajor that contains the major names and the average GPAs per major. Then list the data in GPAbyMajor.

(2 marks)

Question 6 (4 marks)

Assume that a file called `invoiceDF` is created in Spark shell. Calling `invoiceDF.printSchema()` returns the following output:

```
root
|-- InvoiceNo: string (nullable = true)
|-- StockCode: string (nullable = true)
|-- Description: string (nullable = true)
|-- Quantity: integer (nullable = true)
|-- InvoiceTimestamp: timestamp (nullable = true)
|-- UnitPrice: double (nullable = true)
|-- CustomerID: double (nullable = true)
|-- Region: string (nullable = true)
```

Present the Scala source code to implement the following queries on `invoiceDF`: (a) return the total number of rows, (b) return the number of unique stock codes, (c) return the average unit price per stock code and (d) returning the number of invoices per customer ID.

End of Examination