Databases Project – Spring 2017

Prof. Anastasia Ailamaki

Team No: 18

Names: Romain Gehrig, Dominique Roduit, Thierry Treyer

Contents

Contents 1

Deliverable 1 2

Assumptions 2

Entity Relationship Schema 2

Schema 2

Description 2

Relational Schema 2

ER schema to Relational schema 2

DDL 3

General Comments 3

Deliverable 2 4

Assumptions 4

Data Loading 4

Query Implementation 4

Query a: 4

Description of logic: 4

SQL statement 4

Interface 4

Design logic Description 4

Screenshots 4

General Comments 4

Deliverable 3 5

Assumptions 5

Query Implementation 5

Query a: 5

Description of logic: 5

SQL statement 5

Query Analysis 5

Selected Queries (and why) 5

Query 1 5

Query 2 5

Query 3 5

Interface 6

General Comments 6

# Deliverable 1

## Assumptions

<In this section write down the assumptions you made about the data. Write a sentence for each assumption you made>

**Story**

We exploded the attributes of this table in many tables, especially for all the fields containing multiples comma separated values. We describe above the replacement of attributes by new tables.

**script, pencils, inks, colors, letters**

These 5 attributes were replaced by 2 new tables :

1. **person**

this table will contain all the persons (artists or editors) who were contained in the 5 attributes above (and in the deleted attribute “editing”, explanation below). It is thus made of the attributes “alias\_id” who refers to a person, to reproduce the fact that sometimes some artists have an alias name. “firstname” and “lastname”.

1. **Participate**

This is the link between person and story to attribute artists to the stories. This single table fully replace the 5 attributes above, each different attribute can be identified by the “role” field whose the type is an enum which express which of the 5 attribute is concerned by the record.

**characters, feature**

2 attributes replaces by 3 tables. On table “hero” which will store the list of all heros, and one intermediate table by attribute which allow to split each comma separated values in many single record linked to the story.

**genre**

This attribute was replaced by two tables : one table “genre” to contain all the genre and avoid duplicata, and an intermediate table “story\_genre” with two identifying relationships, in order to explode the separated values in multiples records. Each genre associated to a story is a record in this table.

**editing**

this attribute was deleted because we can retrieve it via the “editing\_id” attribute linked to its indicia\_publisher, itself link to the issue containing the story. The editor is now in a separated table “edit” containing only editors and linking a person (an editor in this case) to the issue containing the story published by the indicia\_publisher linked to the concerned story. The credits are stored in the “note” attribute of this table.

**Issue**

**valid\_isbn**

this attribute was removed, we only kept “isbn”.

**price**

attribute replaced by a new table “price” containing the issue concerned by the price, the amount (price itself) and the currency in which this price is exprimed (currency is simply an enum, since the values of currency are a finite set of same values).

## Entity Relationship Schema

<In this section you should have figure of the ER schema as well as descriptions about entities and relations>

### Schema



### Description <Describe all the choices you made for Entities and Relationships>

**Story (relations)**

**story.id - story\_reprint.origin\_id**

An origin story (old printed story) is 0 or one story.

A story can be the origin story 0, 1 or many times (if reprinted many times).

Identifying relationship because the origin story printing is itself a story.

**story.id - story\_reprint.target\_id**

A target story (old printed story) is exactly one story.

A story can be the target story 1 or many times (if reprinted many times).

Identifying relationship because the target story printing is itself a story.

**story.type\_id - story\_type.id**

A story can be of zero or one type.

A story type could be attributed to 0 or many stories.

**story.issue\_id – issue.id**

A story is contained by 1 issue

An issue contains 1 or many issues.

**story.id – participate.story\_id**

A person can participate in one or many stories

A story can have one or many artists who participated to its realisation

**story.id – feature.story\_id**

A feature can be part of zero or one story

A story can have none or many feature.

**story.id – character.story\_id**

A character can be attributed to none or many stories

A story have at least one characters, or many.

**story.id – story\_genre.story\_id**

A genre can be attributed to 1 or many stories

A story can be of one or many genre

**Indicia\_publisher.id – issue.indicia\_publisher\_id**

A story can be published by 0 or 1 indicia publisher

An indicia publisher can publish 1 or many stories

**Issue (relations)**

**Issue\_reprint.origin\_id – issue.id**

An origin issue (old printed issue) is exactly one issue.

An issue can be the origin issue 1 or many times (if reprinted many times).

Identifying relationship because the origin issue is itself an issue.

**Issue\_reprint.target\_id – issue.id**

A target (new printed issue) is exactly one issue.

An issue can be the target issue 1 or many times (if reprinted many times).

Identifying relationship because the target issue is itself an issue.

**Indicia\_publisher.id – issue.indicia\_publisher\_id**

An indicia\_publisher can publish 1 or many issue.

An issue can be published by 0 or one indicia publisher

**issue.id – price.issue\_id**

An issue can be sold to 1 or many price

An a price with its currency can be attributed to only one issue.

**publisher (relations)**

**Publisher.id – indicia\_publisher.publisher\_id**

An indicia publisher can be owned by only one publisher.

A publisher can be the owner of 0 or many indicia publishers.

Identifying relationship because an indicia\_publisher depends directly of a publisher, it can't exists without a publisher.

**Publisher.id – brand\_group.publisher\_id**

A brand is owned by exactly one publisher.

A publisher can own 0 or many brands.

Identifying relationship because an brand\_group depends directly of a publisher, it can't exists without a publisher.

**Publisher.id – series.publisher\_id**

A publisher can publish 0, one or many series.

A serie can be published by one publisher.

**Country (relations)**

**Indicia\_publisher.country\_id – country.id**

An indicia publisher comes from one country

A country can be attributed to 0 or many company

**Country.id – publisher.country\_id**

A publisher comes from one country.

A country can be assign to many publisher.

**Series.country\_id – country.id**

A country can be attributed to 0, one or many series.

A serie comes from one country.

**Language (relations)**

**Language.id – series.language\_id**

A language can be attributed to 0, one or many series.

A series can be in one language.

**Series (relations)**

**Series\_publication\_type.id – series.publication\_type\_id**

A publication type can be attributed to 0, one or many series.

A serie can be of one publication type.

**Series.id – issue.series\_id**

An issue belongs to 0 or one serie.

A serie can wrap 1 or many issues.

**Issue.id – series.first\_issue\_id**

It could exists 0 or 1 first issue for a serie

An issue can be the first issue of 0, 1 or many series.

**Issue.id – series.last\_issue\_id**

It could exists 0 or 1 last issue for a serie

An issue can be the last issue of 0, 1 or many series.

## Relational Schema

### ER schema to Relational schema

<Describe the transition from ER schema to Relational schema>



The relations made only of single strokes as above (thin or bold but without arrows), generate the creation of a new table containing two foreign keys who link together the two tables concerned.

C:\Users\Dominique\AppData\Local\Microsoft\Windows\INetCache\Content.Word\notnullrelation.png

The relations who contains one arrow are reduced to a NON NULL foreign key in the table in front of the arrow.

In the example above, the relation could be translated by a NON NULL foreign key in the table B.

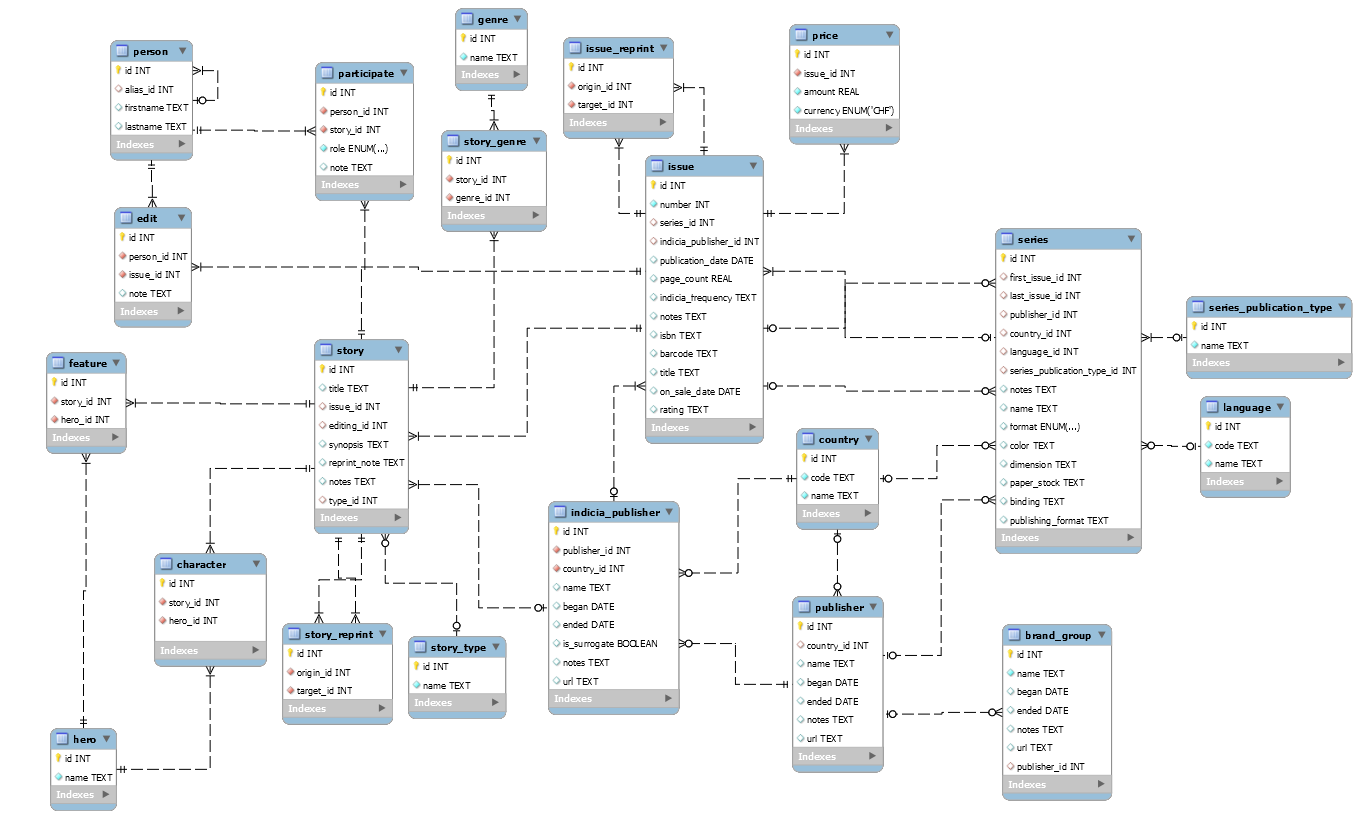


In the relations who are only made of one-one (two bold arrows) and zero-one (one bold and one thin arrows as in this example) constraints, the foreign key must be UNIQUE.

In the example above, the relation could be translated by a UNIQUE foreign key in the table A.

The participation constraints “at least one” (denoted by a bold stroke) can’t be captured in SQL without check constraint or assertion.

### 



### DDL

<Provide the DDL>

-- -----------------------------------------------------

-- Table story\_type

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS story\_type (

id INT NOT NULL AUTO\_INCREMENT,

name TEXT NOT NULL,

PRIMARY KEY (id)

);

-- -----------------------------------------------------

-- Table country

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS country (

id INT NOT NULL AUTO\_INCREMENT,

code TEXT NOT NULL,

name TEXT NOT NULL,

PRIMARY KEY (id)

);

-- -----------------------------------------------------

-- Table publisher

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS publisher (

id INT NOT NULL AUTO\_INCREMENT,

country\_id INT NULL,

name TEXT NULL,

began DATE NULL ,

ended DATE NULL,

notes TEXT NULL,

url TEXT NULL,

PRIMARY KEY (id),

INDEX country\_country\_id\_idx (country\_id ASC),

CONSTRAINT country\_country\_id

FOREIGN KEY (country\_id)

REFERENCES country (id)

ON DELETE SET NULL

);

-- -----------------------------------------------------

-- Table indicia\_publisher

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS indicia\_publisher (

id INT NOT NULL AUTO\_INCREMENT,

publisher\_id INT NOT NULL,

country\_id INT NOT NULL,

name TEXT NULL,

began DATE NULL,

ended DATE NULL,

is\_surrogate TINYINT(1) NULL,

notes TEXT NULL,

url TEXT NULL,

PRIMARY KEY (id),

INDEX country\_country\_id\_idx (country\_id ASC),

INDEX publisher\_publisher\_id\_idx (publisher\_id ASC),

CONSTRAINT country\_country\_id

FOREIGN KEY (country\_id)

REFERENCES country (id),

CONSTRAINT publisher\_publisher\_id

FOREIGN KEY (publisher\_id)

REFERENCES publisher (id)

);

-- -----------------------------------------------------

-- Table language

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS language (

id INT NOT NULL AUTO\_INCREMENT,

code TEXT NOT NULL,

name TEXT NOT NULL,

PRIMARY KEY (id)

);

-- -----------------------------------------------------

-- Table series\_publication\_type

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS series\_publication\_type (

id INT NOT NULL AUTO\_INCREMENT,

name TEXT NOT NULL,

PRIMARY KEY (id)

);

-- -----------------------------------------------------

-- Table series

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS series (

id INT NOT NULL AUTO\_INCREMENT,

first\_issue\_id INT NULL,

last\_issue\_id INT NULL,

publisher\_id INT NULL,

country\_id INT NULL,

language\_id INT NULL,

series\_publication\_type\_id INT NULL,

notes TEXT NULL,

name TEXT NULL,

format ENUM('Hardcover', 'Comics') NULL COMMENT 'Description of physical format (Hardcover, comics, etc.)',

color TEXT NULL,

dimension TEXT NULL,

paper\_stock TEXT NULL,

binding TEXT NULL,

publishing\_format TEXT NULL COMMENT 'UNKNOWN FIELD IN THE WIKI',

PRIMARY KEY (id),

INDEX issue\_first\_issue\_id\_idx (first\_issue\_id ASC),

INDEX issue\_last\_issue\_id\_idx (last\_issue\_id ASC),

INDEX country\_country\_id\_idx (country\_id ASC),

INDEX language\_language\_id\_idx (language\_id ASC),

INDEX publisher\_publisher\_id\_idx (publisher\_id ASC),

INDEX fk\_series\_series\_publication\_type1\_idx (series\_publication\_type\_id ASC),

CONSTRAINT issue\_first\_issue\_id

FOREIGN KEY (first\_issue\_id)

REFERENCES issue (id),

CONSTRAINT issue\_last\_issue\_id

FOREIGN KEY (last\_issue\_id)

REFERENCES issue (id),

CONSTRAINT country\_country\_id

FOREIGN KEY (country\_id)

REFERENCES country (id),

CONSTRAINT language\_language\_id

FOREIGN KEY (language\_id)

REFERENCES language (id),

CONSTRAINT publisher\_publisher\_id

FOREIGN KEY (publisher\_id)

REFERENCES publisher (id),

CONSTRAINT fk\_series\_series\_publication\_type1

FOREIGN KEY (series\_publication\_type\_id)

REFERENCES series\_publication\_type (id)

);

-- -----------------------------------------------------

-- Table issue

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS issue (

id INT NOT NULL AUTO\_INCREMENT,

number INT NOT NULL COMMENT 'Issue number',

series\_id INT NULL,

indicia\_publisher\_id INT NULL,

publication\_date DATE NULL COMMENT 'May be approximate (circa 1870) or a precise date',

page\_count REAL NULL COMMENT 'Is a real number!?!',

indicia\_frequency TEXT NULL COMMENT 'Publication frequency (monthly, quarterly, etc.)',

notes TEXT NULL COMMENT 'Freeform notes',

isbn TEXT NULL COMMENT 'UNKNOWN FIELD IN WIKI',

barcode TEXT NULL,

title TEXT NULL,

on\_sale\_date DATE NULL COMMENT 'UNKNOWN FIELD IN WIKI',

rating TEXT NULL COMMENT 'Approval and age rating',

PRIMARY KEY (id),

INDEX indica\_publisher\_id\_idx (indicia\_publisher\_id ASC),

INDEX series\_id\_id\_idx (series\_id ASC),

CONSTRAINT indica\_publisher\_id

FOREIGN KEY (indicia\_publisher\_id)

REFERENCES indicia\_publisher (id),

CONSTRAINT series\_id\_id

FOREIGN KEY (series\_id)

REFERENCES series (id)

);

-- -----------------------------------------------------

-- Table story

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS story (

id INT NOT NULL AUTO\_INCREMENT,

title TEXT NULL COMMENT 'Name of the story',

issue\_id INT NULL,

editing\_id INT NULL COMMENT 'Is NULL if the editor is the same from the issue',

synopsis TEXT NULL,

reprint\_note TEXT NULL,

notes TEXT NULL COMMENT 'Arbitrary notes',

type\_id INT NULL,

PRIMARY KEY (id),

INDEX story\_type\_type\_id\_idx (type\_id ASC),

INDEX issue\_issue\_id\_idx (issue\_id ASC),

INDEX indicia\_publisher\_id1\_idx (editing\_id ASC),

CONSTRAINT story\_type\_type\_id

FOREIGN KEY (type\_id)

REFERENCES story\_type (id),

CONSTRAINT issue\_issue\_id

FOREIGN KEY (issue\_id)

REFERENCES issue (id),

CONSTRAINT indicia\_publisher\_id1

FOREIGN KEY (editing\_id)

REFERENCES indicia\_publisher (id)

);

-- -----------------------------------------------------

-- Table story\_reprint

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS story\_reprint (

id INT NOT NULL AUTO\_INCREMENT,

origin\_id INT NOT NULL,

target\_id INT NOT NULL,

PRIMARY KEY (id),

INDEX story\_origin\_id\_idx (origin\_id ASC),

INDEX story\_target\_id\_idx (target\_id ASC),

CONSTRAINT story\_origin\_id

FOREIGN KEY (origin\_id)

REFERENCES story (id),

CONSTRAINT story\_target\_id

FOREIGN KEY (target\_id)

REFERENCES story (id)

);

-- -----------------------------------------------------

-- Table issue\_reprint

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS issue\_reprint (

id INT NOT NULL AUTO\_INCREMENT,

origin\_id INT NOT NULL,

target\_id INT NOT NULL,

PRIMARY KEY (id),

INDEX issue\_origin\_id\_idx (origin\_id ASC),

INDEX issue\_target\_id\_idx (target\_id ASC),

CONSTRAINT issue\_origin\_id

FOREIGN KEY (origin\_id)

REFERENCES issue (id),

CONSTRAINT issue\_target\_id

FOREIGN KEY (target\_id)

REFERENCES issue (id)

);

-- -----------------------------------------------------

-- Table brand\_group

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS brand\_group (

id INT NOT NULL AUTO\_INCREMENT,

name TEXT NOT NULL,

began DATE NULL,

ended DATE NULL,

notes TEXT NULL,

url TEXT NULL,

publisher\_id INT NULL,

PRIMARY KEY (id),

INDEX publisher\_publisher\_id\_idx (publisher\_id ASC),

CONSTRAINT publisher\_publisher\_id

FOREIGN KEY (publisher\_id)

REFERENCES publisher (id)

);

-- -----------------------------------------------------

-- Table hero

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS hero (

id INT NOT NULL AUTO\_INCREMENT,

name TEXT NOT NULL,

PRIMARY KEY (id)

);

-- -----------------------------------------------------

-- Table feature

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS feature (

id INT NOT NULL AUTO\_INCREMENT,

story\_id INT NOT NULL,

hero\_id INT NOT NULL,

INDEX story\_id1\_idx (story\_id ASC),

INDEX hero\_id1\_idx (hero\_id ASC),

PRIMARY KEY (id),

CONSTRAINT hero\_id1

FOREIGN KEY (hero\_id)

REFERENCES hero (id),

CONSTRAINT story\_id1

FOREIGN KEY (story\_id)

REFERENCES story (id)

);

-- -----------------------------------------------------

-- Table person

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS person (

id INT NOT NULL AUTO\_INCREMENT,

alias\_id INT NULL,

firstname TEXT NULL,

lastname TEXT NULL,

PRIMARY KEY (id),

INDEX fk\_person\_person1\_idx (alias\_id ASC),

CONSTRAINT fk\_person\_person1

FOREIGN KEY (alias\_id)

REFERENCES person (id)

);

-- -----------------------------------------------------

-- Table character

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS character (

id INT NOT NULL AUTO\_INCREMENT,

story\_id INT NOT NULL,

hero\_id INT NOT NULL,

INDEX story\_id1\_idx (story\_id ASC),

INDEX hero\_id1\_idx (hero\_id ASC),

PRIMARY KEY (id),

CONSTRAINT hero\_id10

FOREIGN KEY (hero\_id)

REFERENCES hero (id),

CONSTRAINT story\_id10

FOREIGN KEY (story\_id)

REFERENCES story (id)

);

-- -----------------------------------------------------

-- Table genre

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS genre (

id INT NOT NULL AUTO\_INCREMENT,

name TEXT NOT NULL,

PRIMARY KEY (id)

);

-- -----------------------------------------------------

-- Table story\_genre

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS story\_genre (

id INT NOT NULL AUTO\_INCREMENT,

story\_id INT NOT NULL,

genre\_id INT NOT NULL,

PRIMARY KEY (id),

INDEX story\_id3\_idx (story\_id ASC),

INDEX genre\_id1\_idx (genre\_id ASC),

CONSTRAINT story\_id3

FOREIGN KEY (story\_id)

REFERENCES story (id),

CONSTRAINT genre\_id1

FOREIGN KEY (genre\_id)

REFERENCES genre (id)

);

-- -----------------------------------------------------

-- Table price

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS price (

id INT NOT NULL AUTO\_INCREMENT,

issue\_id INT NOT NULL,

amount REAL NOT NULL,

currency ENUM('CHF') NOT NULL,

PRIMARY KEY (id),

INDEX issue\_id1\_idx (issue\_id ASC),

CONSTRAINT issue\_id1

FOREIGN KEY (issue\_id)

REFERENCES issue (id)

);

-- -----------------------------------------------------

-- Table participate

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS participate (

id INT NOT NULL,

person\_id INT NOT NULL,

story\_id INT NOT NULL,

role ENUM('script', 'pencil', 'ink', 'color', 'letter') NOT NULL,

note TEXT NULL,

PRIMARY KEY (id),

INDEX person\_id2\_idx (person\_id ASC),

INDEX story\_id4\_idx (story\_id ASC),

CONSTRAINT person\_id2

FOREIGN KEY (person\_id)

REFERENCES person (id),

CONSTRAINT story\_id4

FOREIGN KEY (story\_id)

REFERENCES story (id)

);

-- -----------------------------------------------------

-- Table edit

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS edit (

id INT NOT NULL AUTO\_INCREMENT,

person\_id INT NOT NULL,

issue\_id INT NOT NULL,

note TEXT NULL,

PRIMARY KEY (id),

INDEX fk\_edit\_person1\_idx (person\_id ASC),

INDEX fk\_edit\_issue1\_idx (issue\_id ASC),

CONSTRAINT fk\_edit\_person1

FOREIGN KEY (person\_id)

REFERENCES person (id),

CONSTRAINT fk\_edit\_issue1

FOREIGN KEY (issue\_id)

REFERENCES issue (id)

);

## General Comments

<In this section write general comments about your deliverable (comments and work allocation between team members>

# Deliverable 2

## Assumptions

<In this section write down the assumptions you made about the data. Write a sentence for each assumption you made>

## Data Loading

## Query Implementation

<For each query>

### Query a:

#### Description of logic:

<What does the query do and how do I decide to solve it>

#### SQL statement

<The SQL statement>

## Interface

### Design logic Description

<Describe the general logic of your design as well as the technology you decided to use>

### Screenshots

<Provide some initial screen shots of your interface>

## General Comments

<In this section write general comments about your deliverable (comments and work allocation between team members>

# Deliverable 3

# Assumptions

<In this section write down the assumptions you made about the data. Write a sentence for each assumption you made>

## Query Implementation

<For each query>

### Query a:

#### Description of logic:

<What does the query do and how do I decide to solve it>

#### SQL statement

<The SQL statement>

## Query Analysis

### Selected Queries (and why)

#### Query 1

<Initial Running time:

Optimized Running time:

Explain the improvement:

Initial plan

Improved plan>

#### Query 2

<Initial Running time:

Optimized Running time:

Explain the improvement:

Initial plan

Improved plan>

#### Query 3

<Initial Running time:

Optimized Running time:

Explain the improvement:

Initial plan

Improved plan>

# Interface

### Design logic Description

<Describe the general logic of your design as well as the technology you decided to use>

### Screenshots

<Provide some initial screen shots of your interface>

# General Comments

<In this section write general comments about your deliverable (comments and work allocation between team members>