Revision
YOU choose the topics!

INFO90002 week 6

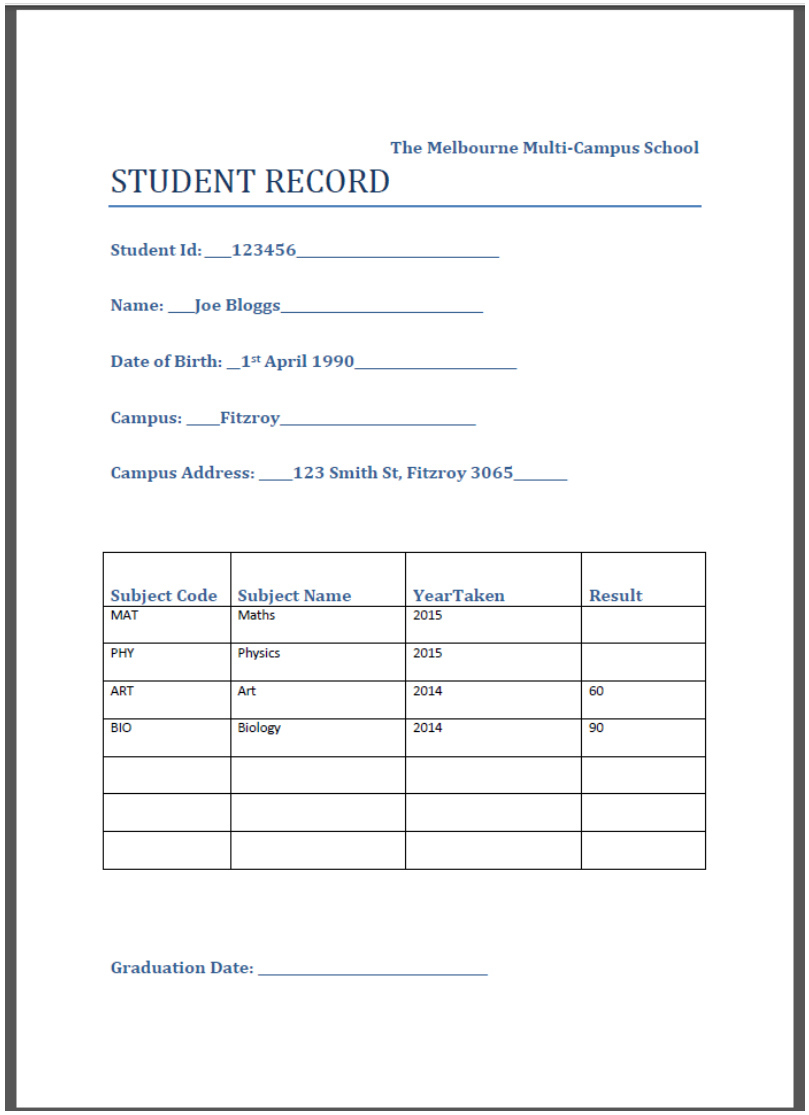
Today's Revision

- Normalization
- Unary relationships and cardinality
- choosing PKs
- data types: Char/Varchar/Numbers/Timestamp
- submitting Asst1: export a PNG
- Asst2: import data into maps

Normalization

- Fundamental Goals
  - Data are stored in tables - each cell has only one value
  - Each fact is only stored once – no repeated data
- Your Guides
  - Functional dependencies
  - Every attribute of a table is dependent on the key, the whole key, and nothing but the key
- Stages
  - 1st normal form – rows and columns, no repeating groups
  - 2nd normal form – non-key attributes determined by ENTIRE key
  - 3rd normal form – non-key attributes determined only by key
  - Higher forms will not be assessed but are worth studying
- May be redundant if you do entity-relationship design

Normalization example



The Melbourne Multi-Campus School

STUDENT RECORD

Student Id: ___123456_____

Name: ___Joe Bloggs_____

Date of Birth: __1st April 1990_____

Campus: ____Fitzroy_____

Campus Address: ____123 Smith St, Fitzroy 3065_____

Subject Code	Subject Name	YearTaken	Result
MAT	Maths	2015	
PHY	Physics	2015	
ART	Art	2014	60
BIO	Biology	2014	90

Graduation Date: _____

- The *Melbourne Multi-Campus School* has been using a paper-based information system.
- Student data is recorded on these sheets of paper.
- Now they are going to computerize.
- How shall we store the data in a relational database?

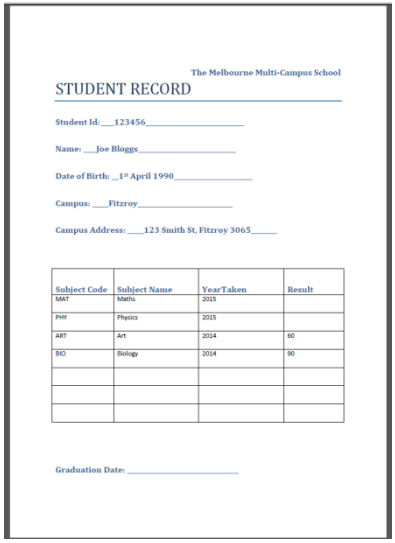
Normalization example

- StudentRecord(studentId, name, dob, campusId, campusAddress,
  (subjectCode, subjectName, yearTaken, result), gradDate)

- StudentRecord(studentId, name, dob, campusId, campusAddress, gradDate)
- StudentTakesSubject(*studentId*, subjectCode, subjectName, yearTaken, result)

- StudentRecord(studentId, name, dob, campusId, campusAddress, gradDate)
- StudentTakesSubject(*studentId*, *subjectCode*, yearTaken, result)
- Subject(subjectCode, subjectName)

- StudentRecord(studentId, name, dob, *campus*, gradDate)
- StudentTakesSubject(*studentId*, *subjectCode*, yearTaken, result)
- Subject(subjectCode, subjectName)
- Campus(campusId, campusAddress)



The Melbourne Multi-Campus School

STUDENT RECORD

Student Id: ___123456_____

Name: ___Joe Bloggs_____

Date of Birth: _1st April 1990_____

Campus: ____Fitzroy_____

Campus Address: ___123 Smith St, Fitzroy 3065_____

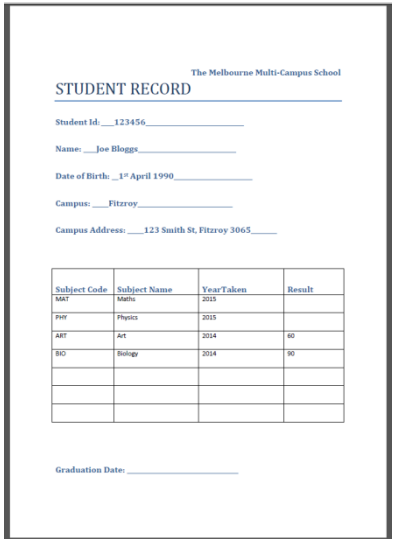
Subject Code	Subject Name	YearTaken	Result
MAT	Maths	2015	
PHY	Physics	2015	
ART	Art	2014	60
BIO	Biology	2014	90

Graduation Date: _____

Update anomolies

if your database is not properly normalized, you may get:

- redundant data (multiple fact recorded >1 times)
- update anomalies
- insert anomalies
- delete anomalies



The Melbourne Multi-Campus School

STUDENT RECORD

Student Id: 123456

Name: Joe Bloggs

Date of Birth: 1st April 1990

Campus: Fitzroy

Campus Address: 123 Smith St, Fitzroy 3065

Subject Code	Subject Name	Year Taken	Result
MAT	Maths	2015	
PHY	Physics	2015	
ART	Art	2014	60
BIO	Biology	2014	90

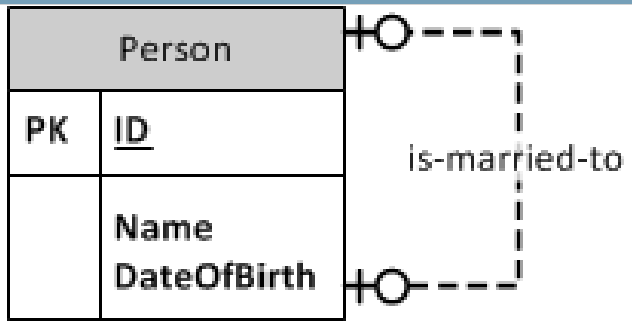
Graduation Date:

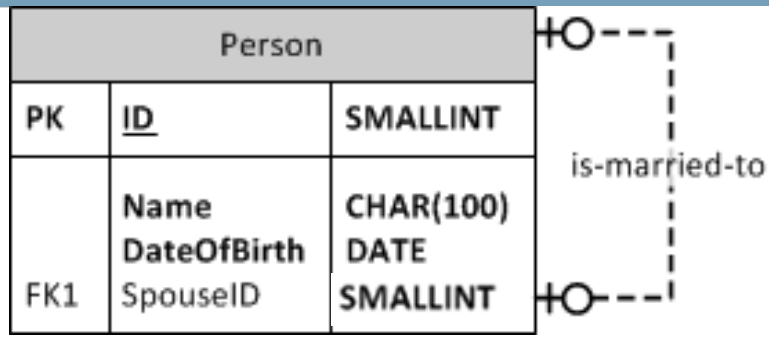
Invoice no	Date	CustNo	CustName	CustAddress	ClerkNo	ClerkName	Terms	ProductNo	ProductDesc	Unit Price
INV0012	14-Aug-09	123	John	128 AA Juanita Ave,	2	Charles Wooten	COD	PS V880.006	AMD Athlon X2DC-	580
INV0012	14-Aug-09	123	John	128 AA Juanita Ave,	2	Charles Wooten	COD	PS.V880.037	PDC E5300	645
INV0012	14-Aug-09	123	John	128 AA Juanita Ave,	2	Charles Wooten	COD	LC.V890.002	LG 18.5" LCD	230
INV0012	14-Aug-09	123	John	128 AA Juanita Ave,	2	Charles Wooten	COD	HP Q754.071	HP LaserJet 5200	1103

Unary Relationships

- Operate in the same way exactly as binary relationships
  - One-to-One
    - put a Foreign key in the entity
  - One-to-Many
    - put a Foreign key in the entity
  - Many-to-Many
    - create an extra table - Associative Entity
    - put two Foreign keys in the Associative Entity
      - the two FKs need different names
      - the FKs become the combined PK of the Associative Entity

Unary – One-to-One





Logical Design

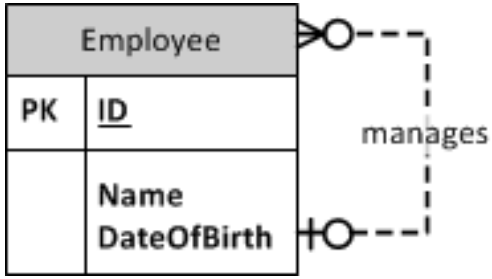
Physical Design

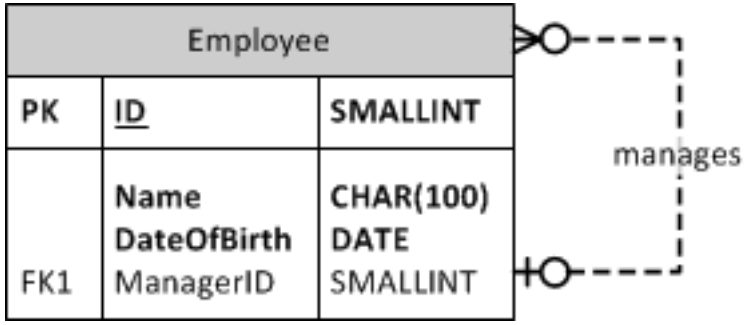
(ID, Name, DateOfBirth, SpouseID)

ID	Name	DOB	SpouseID
1	Ann	1969-06-12	3
2	Fred	1971-05-09	
3	Chon	1982-02-10	1
4	Nancy	1991-01-01	

```
CREATE TABLE Person (
    ID                  smallint,
    Name                varchar(150)  NOT NULL,
    DateOfBirth         DATE          NOT NULL,
    SpouseID            smallint      NOT NULL,
    PRIMARY KEY  (ID),
    FOREIGN KEY (SpouseID) REFERENCES Person(ID)
          ON DELETE RESTRICT
          ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Unary – One-to-Many





Logical Design

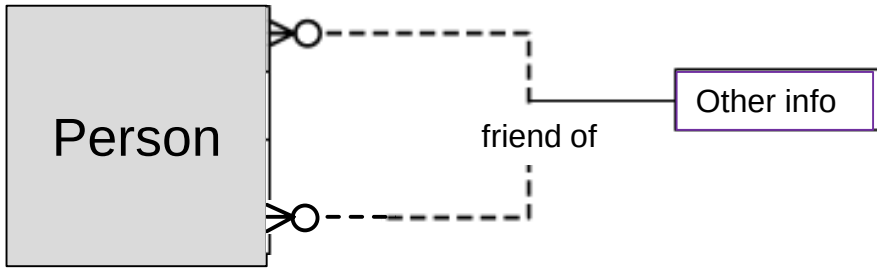
Physical Design

(ID, Name, DateOfBirth, ManagerID)

ID	Name	DOB	MngrID
1	Ann	1969-06-12	
2	Fred	1971-05-09	1
3	Chon	1982-02-10	1
4	Nancy	1991-01-01	1

```
CREATE TABLE Employee (
    ID                  smallint,
    Name                varchar(150)  NOT NULL,
    DateOfBirth         DATE          NOT NULL,
    ManagerID           smallint      NOT NULL,
    PRIMARY KEY  (ID),
    FOREIGN KEY (ManagerID) REFERENCES Employee(ID)
          ON DELETE RESTRICT
          ON UPDATE CASCADE
) ENGINE=InnoDB;
```

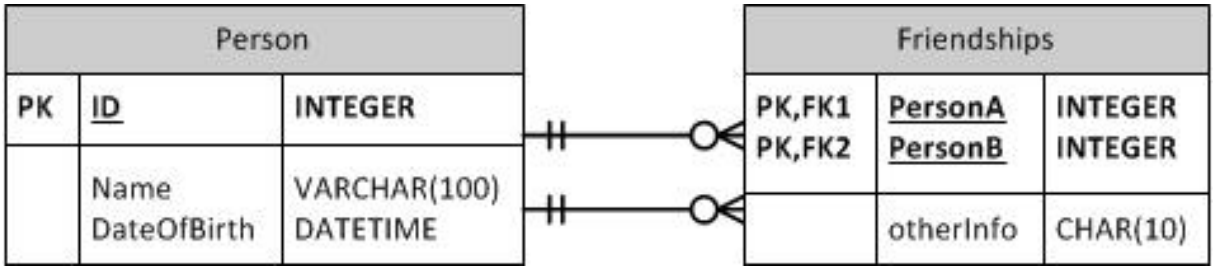
Unary – Many-to-Many



- Logical Design
  - Set up Associative Entity as for any M-M relationship
    - Person = (ID, Name, DateOfBirth)
    - Friendship = (PersonA, PersonB, otherInfo)

Unary – Many-to-Many

- Physical Design



- Implementation

```
-- -----------------------------------------------------
-- Table `mydb`.`Person`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Person` (
  `ID` INT NOT NULL,
  `Name` VARCHAR(50) NULL,
  `DateOfBirth` DATE NULL,
  PRIMARY KEY (`ID`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Friendship`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Friendship` (
  `PersonA` INT NOT NULL,
  `PersonB` INT NOT NULL,
  `otherInfo` CHAR(10) NULL,
  PRIMARY KEY (`PersonA`, `PersonB`),
  INDEX `fk_Friendship_Person1_idx` (`PersonB` ASC),
  CONSTRAINT `fk_Friendship_Person`
    FOREIGN KEY (`PersonA`)
    REFERENCES `mydb`.`Person` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Friendship_Person1`
    FOREIGN KEY (`PersonB`)
    REFERENCES `mydb`.`Person` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Character types (MySQL)

- **CHAR(M):** A fixed-length string that is always right-padded with spaces to the specified length when stored on disc. The range of M is 1 to 255.
- **CHAR:** Synonym for CHAR(1).
- **VARCHAR(M):** A variable-length string. Only the characters inserted are stored – no padding. The range of M is 1 to 65535 characters.
- **BLOB, TEXT:** A binary or text object with a maximum length of 65535 (2^16) bytes (blob) or characters (text). Not stored inline with row data.
- **LONGBLOB, LONGTEXT:** A BLOB or TEXT column with a maximum length of 4,294,967,295 (2^32 - 1) characters.
- **ENUM** (‘value1’,’value2’,…) up to 65,535 members.

Number types (MySQL)

- Integers
  - **TINYINT**: Signed (-128 to 127), Unsigned (0 to 255)
  - **BIT**, **BOOL:** synonyms for TINYINT
  - **SMALLINT**:
    Signed (-32,768 to 32,767), Unsigned (0 to 65,535 – 64k)
  - **MEDIUMINT**:
    Signed (-8388608 to 8388607), Unsigned (0 to 16777215 –16M)
  - **INT / INTEGER**:
    Signed (-2,147,483,648 to 2,147,483,647),
    Unsigned (0 to 4,294,967,295 – 4G or 2^32)
  - **BIGINT**:
    Signed (-9223372036854775808 to 9223372036854775807),
    Unsigned (0 to 18,446,744,073,709,551,615 - 2^64)
  - **Don't** use the "(M)" number for integers

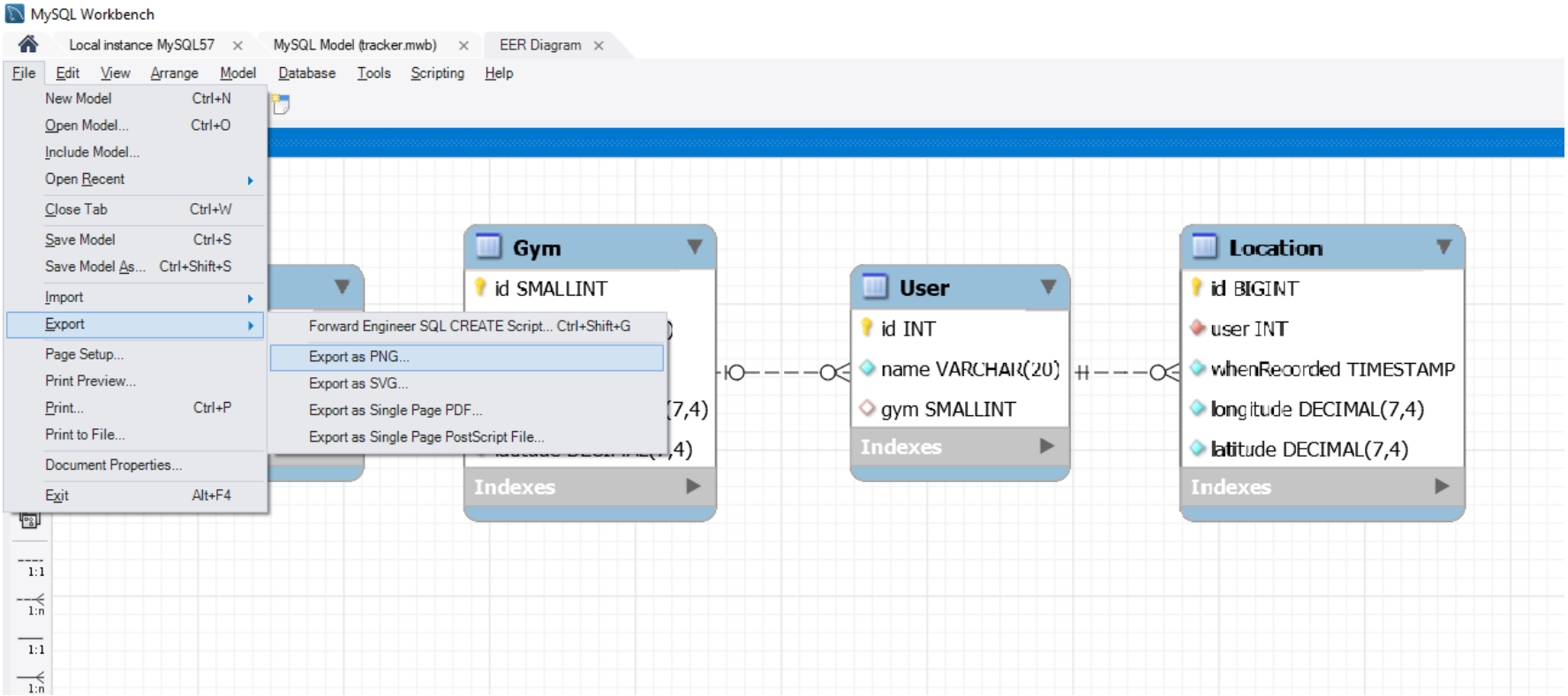
Number types (MySQL)

- Real numbers (fractions)
  - **FLOAT**: single-precision floating point, allowable values: -3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38.
  - **DOUBLE / REAL**: double-precision, allowable values: -1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308.
  - optional M = number of digits stored, D = number of decimals.
  - Float and Double are often used for scientific data.
  - **DECIMAL**[(M[,D])]: fixed-point type. Good for money values.
  - M = precision (number of digits stored), D = number of decimals

Time types

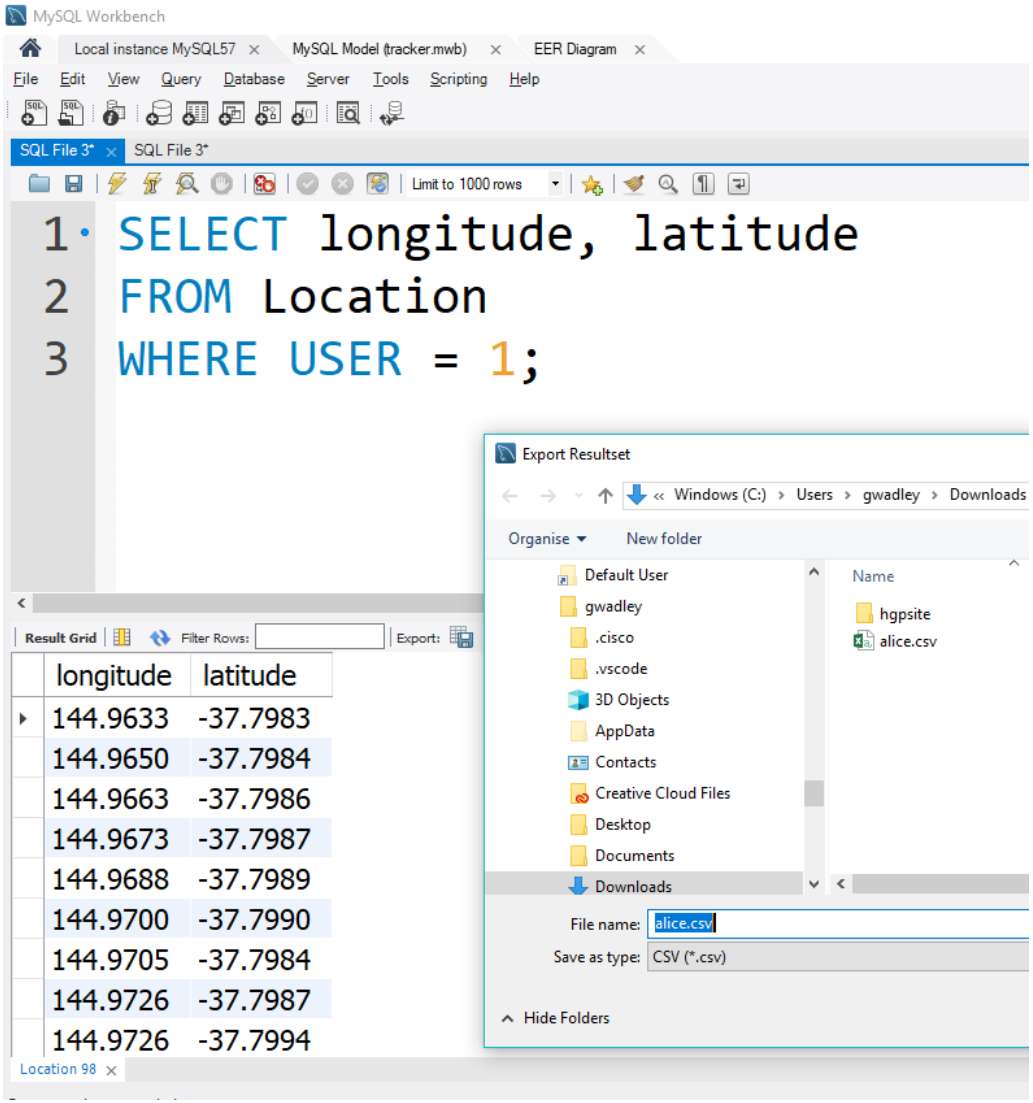
- **DATE** 1000-01-01 to 9999-12-31
- **TIME** -838:59:59 to 838:59:59
  (time of day or elapsed time)
- **DATETIME** 1000-01-01 00:00:00 to
  9999-12-31 23:59:59
- **TIMESTAMP** 1970-01-01 00:00:00 - ~ 2037
  Stored in UTC, converted to local
- **YEAR** 1901 to 2155

Exporting your model to PNG to PDF



- don’t screenshot – export as PNG, then include in your PDF

Location data -> Excel -> Google Maps



- Want to visualize your location data?
- step 1: export from MySQL to CSV
- step 2: import into Google My Maps