COMP . . .

A little template for cheatsheets to come!

Sample question 1

CREATE Syntax

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]
    ↪ tbl_name
    (create_definition,...)
    [table_options]
```

create_definition:

```
col_name column_definition
| [CONSTRAINT [symbol]] PRIMARY KEY [
    ↪ index_type] (index_col_name,...)
[index_option] ...
| {INDEX|KEY} [index_name] [index_type] (
    ↪ index_col_name,...)
[index_option] ...
| [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY]
[index_name] [index_type] (index_col_name
    ↪ ,...)
[index_option] ...
| {FULLTEXT|SPATIAL} [INDEX|KEY] [index_name
    ↪ ] (index_col_name,...)
[index_option] ...
| [CONSTRAINT [symbol]] FOREIGN KEY
[index_name] (index_col_name,...)
    ↪ reference_definition
| CHECK (expr)
```

column_definition:

```
data_type [NOT NULL | NULL] [DEFAULT
    ↪ default_value]
[AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY]
    ↪ KEY]
[COMMENT 'string']
[COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}]
[STORAGE {DISK|MEMORY|DEFAULT}]
[reference_definition]
```

Create Table Using Another Table

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

Alter table Syntax

```
# add a column
alter table tablename
add column_name datatype
# drop a column
alter table tablename
drop col1, drop col2
# alter a column's␣datatype
alter␣table␣tablename
alter␣column␣column_name␣datatype␣/␣modify␣
    ↪ column␣column_name␣datatype␣
    ↪ auto_increment
#␣add␣a␣constraint
alter␣table␣tablename
add␣constraint␣fk␣foreign␣key␣(col)␣
    ↪ references␣tblname2(col2)
```

Update Syntax

```
UPDATE table1
SET table1col1 = 1
FROM table2
WHERE col2 = 2
```

Create index Syntax

```
create index idx_name
on table_name(col)
```

Constraint Syntax

Sample Solution 1

2.

```
create table addresses (
id int auto_increment unique not null,
street_name varchar(100) not null,
city varchar(20) not null,
region varchar(20),
postal_code int(5),
country varchar(20) not null,
constraint uc_address unique(
    ↪ street_name, city, region,
    ↪ postal_code, country),
constraint pk_addresses primary key(id)
)
```

3.

```
insert addresses(addresses, city,
    ↪ region, postal_code, country)
        select addresses, city, region,
            ↪ postal_code, country
    from customers
    group by addresses, city, region,
        ↪ postal_code, country
OR
insert into addresses(...)
    select distinct address, city,
        ↪ region, postal_code, country
        ↪ from customers
```

4.

```
# Clone a table
create table new_customers like
    ↪ customers
# Load the data
insert into new_customers
select * from customers
```

5.

```
alter table new_customers
add column address_id int
```

6.

```
update table new_customers
set new_customers.address_id =
    ↪ addresses.id
from new_customers join addresses on
new_customers.address = addresses.
    ↪ street_name
and new_customers.city = addresses.city
and new_customers.region = addresses.
    ↪ region
and new_customers.postal_code =
    ↪ addresses.postal_code
and new_customers.country = addresses.
    ↪ country
OR
```

```
‘1upda ‘Q1E4RTYU7IOP[-\te new_customers
set address_id=(select id from address
    ↪ where
(...AND (... OR (adrresses.region is
    ↪ null or new_customers.region os
    ↪ null)))
```

7.
```
alter table addresses
drop column street_name,
drop city,
drop region,
drop postal_code,
drop country
```

8.

```
alter table new_customers
add constraint
    ↪ fk_new_customers_addresses
    ↪ foreign key (address_id)
references addresses(id);
create index idx_new_customer_id on
new_customers(id)
```

Sample question 2

Define a datamodel using the notation we used in class. You should define either a logical or physical model. From your model, create the SQL DDL. The datamodel is the following.

- Companies: These are businesses and have properties Name, ID. The ID should be unique and derived from the company name. Companies also have an address.

- Persons: A Person has a last_name, first_name and middle initial. A Person also has an email. A Person also has an address.

- A Company may be related to one or more Persons. A Person may be related to one or more Companies. A Person-Company relationship has a type, e.g. Employee, Contractor, Consultant. The data model should ONLY allow creation of relationships that conform to one of the types. It must be possible to add new, named types.