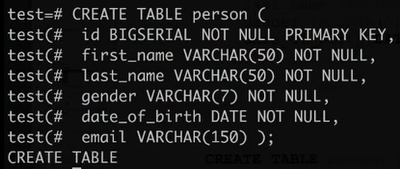
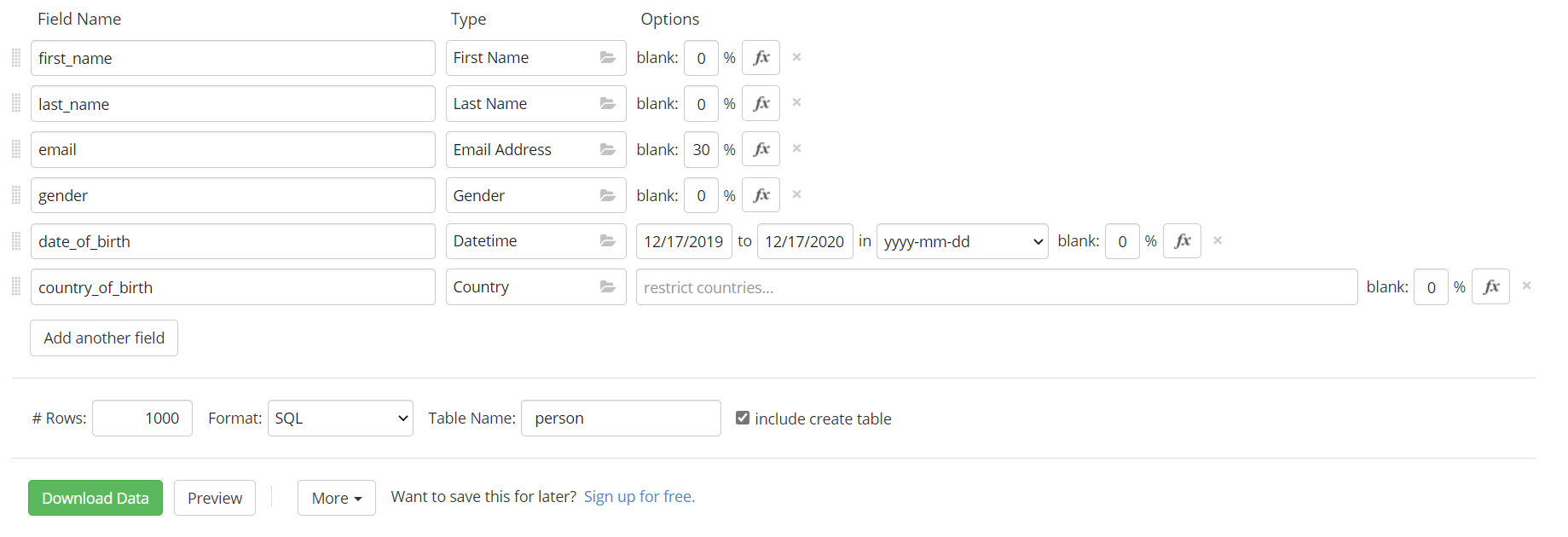
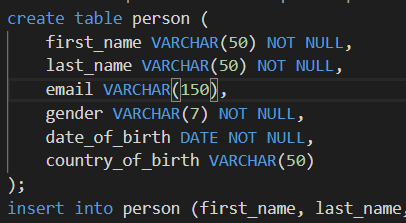
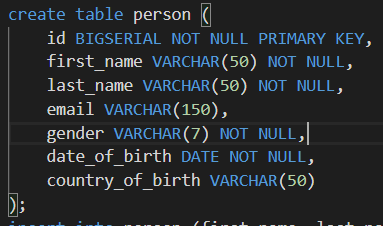
* [SQL - A Quick Overview |¦| SQL Tutorial |¦| SQL for Beginners - YouTube](https://www.youtube.com/watch?v=OfM5lC-7R4Y&list=PLih4ch-U2DiBbMoFK4ML9faT3k3MM2UQY&index=2)
  + Collection of tables in the database is called schema
  + 3000: more SQL Commands to create tables
  + 4000: how to insert values into the table … also update, delete, display, alter, drop, select, insert, delete
* [Relational Databases - How to Choose |¦| SQL Tutorial |¦| SQL for Beginners - YouTube](https://www.youtube.com/watch?v=WzfDLqt-WIg&list=PLih4ch-U2DiBbMoFK4ML9faT3k3MM2UQY&index=3)
* [PostgreSQL (Postgres) - Installation & Overview |¦| SQL Tutorial |¦| SQL for Beginners - YouTube](https://www.youtube.com/watch?v=fZQI7nBu32M)
  + 6000: Side Menu Bar: like postgres create another database (under Database)
* [Intro To PostgreSQL Databases With PgAdmin For Beginners - Full Course - YouTube](https://www.youtube.com/watch?v=Dd2ej-QKrWY)
  + Rows are Records and Columns are Fields = a DB is a collection of tables
  + under Database the postgres Database comes by default
  + 2100: starts the tutorial, I have followed along
  + Schemas/public/tables
  + Datatype: **bigserial** … ideal for IDs, counts step by step, increment by themself
  + **Not NULL?** It is impossible, that this is empty: useful at IDs
  + Table/customers/ViesEditData/AllRows … gives the QueryTool
  + **Character varying** = varchar
  + **Import Data:** go to table and import/turn on import/file path
* **Entries by Hand**
  + - GUI: Table/ViewEditData/enter in the fields of the GUI/(MenuBar): SaveDataChanges
    - QueryTool: customers/querytool/type in SQL commands … puts in the values into the table – via insert into: columns to fill are given (if one entry is missing, then error)
      * INSERT INTO public."customers"("name", "email")

VALUES ('Harry Brown', 'harry@brown.com')

* **Other SQL Statements:**
  + - Select everything (what will be shown as a result of the query)
      * SELECT \* FROM public.customers
    - Only name
      * SELECT name FROM public.customers
    - Select more columns
      * SELECT name, email FROM public.customers
    - Get Columns under another name
      * SELECT name "myCustomers" FROM public.customers
  + **Where clause:** works very well with if clauses
    - Get person with id=3
      * SELECT \* FROM public.customers

WHERE id=3

* + - Get all without id=3, greater, equals
      * WHERE id!=3
      * WHERE id>3
      * WHERE id>=3
    - Search name: be very strict with the pronunciation/typo/bigSmall
      * WHERE name LIKE 'Tina Smith'
  + **Wildcard %:** typo alert
    - Get everything:
      * WHERE name LIKE '%'
    - Get anything which has Tina at the beginning
      * WHERE name LIKE 'Tina%'
    - Anything with the ending smith
      * WHERE name LIKE '%Smith'
    - Anything with in between
      * WHERE name LIKE '%hn%'
      * WHERE email LIKE '%@%'
  + **AND OR NOT:** Comparison Operator: combine more conditions in the where clause
    - Both conditions have to be true
      * WHERE email LIKE '%hn%' AND id = 4
    - One or the other can be true
      * WHERE name LIKE '%Smith%' OR id = 4
    - Will return anything but “smith”
      * WHERE NOT name LIKE '%Smith%'
  + **Order by:** ORDER BY name ASC(1-5)/DESC(5-1)
    - Orders the names alphabetically
      * ORDER BY name ASC
    - Orders numerically
      * ORDER BY id DESC
* [Learn PostgreSQL Tutorial - Full Course for Beginners - YouTube](https://www.youtube.com/watch?v=qw--VYLpxG4)
  + Software: psql
  + 3000: Database Introduction
    - SQL as programming language:
      * SELECT columnName FROM tableName
  + Date is stored in tables with columns and rows
  + Relational Database: there is a connection to the other tables
  + The PC is the server and can connect to it
  + **CMD is more preferred**
* **SQL COMMANDS FOR BASIC SETUP**
  + **Setup PSQL (Windows):**
    - open SQL Shell: enter until password – enter password (will not show letters)
    - by default, you get the database called “postgres”
    - you have to be connected to your database
    - connect to another database: Database [postgres]: test //ERROR, because it does not exist
  + Create Database: postgres=#: this has to appear to do these commands
    - \q: means quit //it quits the psql mode – but for me in closes the shell
    - help: for felp
    - postgres=# \?: more help
    - just press q: you will go to this > postgres=#
    - postgres=# \l: list of all Databases
  + **Create a new Database:** 
    - postgres=# CREATE DATABASE test; //without semicolon no execution
  + **how to connect to a database:** 
    - psql help: gives more
    - add then do this: psql -h localhost -p 5432 – U username test
      * but this does not work somehow: therefore restart shell and connect “normally”
    - test=# \c test …. Another way to connect to the database
  + **delete database:** 
    - DROP DATABASE test; //dangerous, deletes all stored data
* **CREATE TABLES**
  + Look Up which datatypes are possible in the documentation
  + Create a table: be aware of typo, comma and datatype, semicolon at the end
    - test=# CREATE TABLE person(
    - test(# id INT,
    - test(# first\_name VARCHAR(50),
    - test(# last\_name VARCHAR(50),
    - test(# gender VARCHAR(7),
    - test(# date\_of\_birth DATE );
  + show all tables
    - test=# \d
  + show the rows and columns of the tables: just mention the name of the table specifically
    - test=# \d person
* **CREATE TABLES WITH CONSTRAINS**
  + Clear terminal:
    - \! cls
  + How to drop, not the single DB, but only table
    - test=# DROP TABLE person;
  + Before entering something, those constraints have to be satisfied
    - 
    - test=# \d … now shows two tables: because of bigserial
* insert records into databas
  + specify the columns and then type in the values in the command
    - test=# INSERT INTO person(
    - test(# first\_name,
    - test(# last\_name,
    - test(# gender,
    - test(# date\_of\_birth)
    - test-# VALUES ('Anne','Smith','FEMALE', DATE '1988-01-09');
      * INSERT 0 1
  + show all tables of the database
    - test=# \dt
* insert arbitrary many people into the table
  + [Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel](https://www.mockaroo.com/)
  + 
  + Download
  + To open the sql file – download vscode
  + After opening it adjust few things: nullable //we have also included a new column
    - 
  + Shell: test=# \? … in the input/output section
    - \i DATEI Befehle aus Datei ausf³hren
    - [import and export PostgreSQL database - YouTube](https://www.youtube.com/watch?v=C30q5i1e9KE) – does work as well, but not for me
    - You can also copy the sql commands from vscode
    - But we forgot id
      * 
* SELECT FROM
  + … gives the number of entires
    - SELECT FROM person;
  + Only get the first names
    - SELECT first\_name FROM person;
  + get first and last name
    - SELECT first\_name, last\_name FROM person;
  + Get the email column: but the first will be empty because Anne hasn’t any
    - SELECT first\_name, last\_name FROM person;
* ORDER BY: ASC vs. DESC
  + Some basics
    - SELECT \* FROM mock\_data ORDER BY county\_of\_birth; //**asc by default**
    - SELECT \* FROM mock\_data ORDER BY county\_of\_birth **asc**; //same as above
    - test=# SELECT \* FROM mock\_data ORDER BY county\_of\_birth **desc**;
    - SELECT \* FROM mock\_data ORDER BY **email desc**; //get all nulls at first
  + Combine: but better to do it only with only one column
    - SELECT \* FROM mock\_data ORDER BY id, email;
* DISTINCT:
  + Some basics
    - SELECT county\_of\_birth FROM mock\_data; //gives only the countries
    - SELECT county\_of\_birth FROM mock\_data ORDER BY county\_of\_birth; //gives only the countries and orders them by alphabet
  + Only get the country only one time in the output:
    - SELECT **DISTINCT** county\_of\_birth FROM mock\_data ORDER BY county\_of\_birth;
* WHERE CLAUSE:
  + Get all data from the gender female
    - SELECT \* FROM mock\_data WHERE gender='Female';
  + Combine Where with AND/OR
    - SELECT \* FROM mock\_data WHERE gender='Female' AND coucounty\_of\_birth='Poland';
    - SELECT \* FROM mock\_data WHERE gender='Female' AND (county\_of\_birth='Poland' OR county\_of\_birth='China'); //all woman from china and Poland
  + More combination
    - SELECT \* FROM mock\_data WHERE gender='Female' AND (county\_of\_birth='Poland' OR county\_of\_birth='China') test-# AND last\_name='Labitt';
* Comparison Operator
  + Some basics
    - SELECT 1=2; … gives you false also possible with >=, < etc.
    - SELECT 1<>2; … is it not equal? This gives you TRUE
  + auf strings
    - SELECT 'hallo' <> 'hallllo'; //gives true
    - SELECT 'hallo'='hallo'; //gives true
* Limit, Offset and Fetch
  + Select only first 10 values
    - SELECT \* FROM mock\_data LIMIT 10;
  + Choose 10 person after 5 entries
    - SELECT \* FROM mock\_data OFFSET 5 LIMIT 10;
    - SELECT \* FROM mock\_data OFFSET 5; //everybody after the fifth person
  + SQL Standard
    - SELECT \* FROM mock\_data OFFSET 5 FETCH FIRST 5 ROW ONLY;
* IN
  + Get everything from china brazil and France: not with triple ANDs
    - SELECT \* FROM mock\_data WHERE county\_of\_birth IN ('China', 'Brazil', 'France'); //takes a the array of values
    - SELECT \* FROM mock\_data WHERE county\_of\_birth IN ('China', 'Brazil', 'France') ORDER BY county\_of\_birth;
* BETWEEN //select from a range
  + Get everything from persons from this date to this date
    - SELECT \* FROM mock\_data WHERE date\_of\_birth BETWEEN DATE '2020-11-24' AND '1920-11-24';
* LIKE … wildcard: followed by, in between, or at the beginning
  + Some basics
    - SELECT \* FROM mock\_data WHERE email like '%.com';
    - SELECT \* FROM mock\_data WHERE email like '%.com%';
    - SELECT \* FROM mock\_data WHERE email like '.com%';
  + Underscore instead of wildcard … has to match single characters
    - SELECT \* FROM MOCK\_DATA WHERE email LIKE '\_\_\_\_\_\_\_@%';
  + ALIKE: more easy with the spelling
    - SELECT \* FROM MOCK\_DATA WHERE county\_of\_birth ILIKE 'p%';
* GROUP BY:
  + The the count() function will count of often the one thing appears in the specific column
    - SELECT county\_of\_birth, count(\*) FROM MOCK\_DATA GROUP BY county\_of\_birth;
* GROUP BY HAVING: extra filtering
  + Get country with at least 5 people
    - SELECT county\_of\_birth, count(\*) FROM MOCK\_DATA GROUP BY county\_of\_birth HAVING COUNT (\*) >= 5 ORDER BY county\_of\_birth;
    - Best way to use more uncommon functions: GOOGLE AGGREGATE FUNTIONS
* MAX, MIN, SUM
  + SELECT MAX(price) FROM car;
  + SELECT MIN (price) FROM car;
  + SELECT AVG(price) FROM car;
  + SELECT ROUND(AVG(price)) FROM car; //get the average rounded
  + SELECT make, model, MIN(price) FROM car GROUP BY make,model; //lists the brand and the model with the minimum price
  + SELECT SUM(price) FROM car; //get the price sum of all cars
  + SELECT make, SUM(price) FROM car GROUP BY make; //list all brand with their price sums
* BASIC ARETHMETHIC OPERATIONS
  + SELECT 10+-\*/2 //basic calculator stuff
  + SELECT 10 % 3; //give the remainder = 1
* ARETHMETIC ROUND
  + More mathematical calculation
* ALIAS
  + Renaming the column name : AS
* COALESCE
  + Set up default value, for nulls
* NULLIF
  + if both arguments are the same it returns NULL, otherwise the first argument
* DATES
  + Look up doku, for us TIMESTAMP
* Subtract Dates
  + Is more like normal mathematic, use INTERVALL
* EXTRACT
  + What do you want to have from the time stamp … EXTRACT ()
* AGE FUNCTION:
  + Can calculate with the date with a reference timestamp

**VERY IMPORTANT FOR THE PROJECT**

* **PRIMARY KEYS:**
  + Everything is the same – how to distinguish? Like the passport\_number, but we use id
  + Just show the table with: \d mock\_data, you will see the Primary Key at the end
    - when you try to inert another person with the same id, ERROR
  + how to drop the constraint? Like delete the id row
    - ALTER TABLE mock\_data DROP CONSTRAINT mock\_data\_pkey;
    - Now a person with the same id can be inserted with:
      * insert into MOCK\_DATA (id, first\_name, last\_name, email, gender, date\_of\_birth, county\_of\_birth) values (1, 'Tucker', 'Segar', null, 'Male', '2020-05-16', 'Brazil');
    - SELECT \* FROM mock\_data WHERE id=1;
      * Now you get two names, but can not be uniquely identified
  + How to add primary key
    - 238: therefore get use ADD, and in brackets put in your primary single primary key or combinations of several primary key
      * ALTER TABLE mock\_data ADD PRIMARY KEY (id); //this does not work now because, there are two persons with the same id, you have to delete both before
    - DELETE FROM mock\_data WHERE id=1; //everybody with the id= 1 will be deleted
      * NOW you can add the primary key constraint:
        + ALTER TABLE mock\_data ADD PRIMARY KEY (id); //now it works because the id is unique in this table
  + Unique Constraints: be able to have unique values for a given column
    - Until now one column (besides primary key) can have the same values, not to misunderstood with the primary key, e.g. you cannot have the same email address
      * ALTER TABLE mock\_data ADD CONSTRAINT unique\_email\_adress UNIQUE (email); //cannot not exe, if there are already multiple entries in the row, which have the same value, therefore can delete the person or change the column value
      * ALTER TABLE mock\_data ADD UNIQUE (email); //automatic name, but the same as the first
    - How to drop constraints
      * ALTER TABLE mock\_data DROP CONSTRAINT unique\_email\_adress;
  + Check Constraints
    - ALTER TABLE mock\_data ADD CONSTRAINT gender\_constraint CHECK (gender = 'Female' OR gender = 'Male'); //does not allow other genders than female and male
  + DELETE RECORDS: best to use ID, but also you can delete after other attributes
    - DELETE FROM mock\_data WHERE first\_name='Tucker';
      * Also combining the Where condition with AND OR is possible
  + UPDATE RECORDS:
    - UPDATE mock\_data SET email ='hahahahah' WHERE id=6;
    - Update more columns at once
      * UPDATE mock\_data SET first\_name = 'Rishad', last\_name = 'Howlader', email= 'technikum' WHERE id=6;
  + CONFLICTS: not allowed to insert in the same column, nut only works if the column have unique constraints
    - Do NOTHING: kind of try/catch
  + Upsert: customer want to insert a renewed email again, although Conflict still update
* FOREIGN KEYS & JOINS is a COLUMN which references a primary Key, Types have to be same
  + One person can only have one car and one car can only belong to one person
    - die class with the deriving primary key has to be created beforehand
* Updating Foreign IDs
  + UPDATE person SET car\_id = 2 WHERE id = 1;
  + UPDATE person SET car\_id = 2 WHERE id=2; //ERROR das auto gehört schon einem, wegen Constraint
* INNER JOINS: combining two tables, takes what ever is common in both tables
  + Combines two tables
    - SELECT \* FROM person

JOIN car on person.car\_id = car.id;

* + LEFT JOINS:
  + RIGHT JOINS:
* DELETING RECORDS WITH FOREIGN KEYS:
  + record cant be deleted if it references
  + cascade can be used to do it either way, but it is bad practice
* CREATE CSV:
  + COME BACK TO IT LATER WITH THE JOINS
* Serial and Sequence: how the BIGINT works
* Extension: “Libraries”, //all will be listed: SELECT \* from pg\_available\_extensions;
* UUID: install extension // CREATE EXTENSION IF NOT EXISTS "uuid-ossp";