



Лабораторная 1

Знакомство с командной строкой Ubuntu и её основные консольные команды

Установка WSL

WSL (Windows Subsystem for Linux) — слой совместимости для запуска Linux-приложений (двоичных исполняемых файлов в формате ELF) в ОС Windows 10 и Windows 11.

Для установки требуется в PowerShell ввести команду: `wsl --install`. Эта команда включит функции необходимые для запуска WSL и установки дистрибутива Ubuntu для Linux.

Для перехода в WSL требуется в PowerShell ввести команду: `wsl`

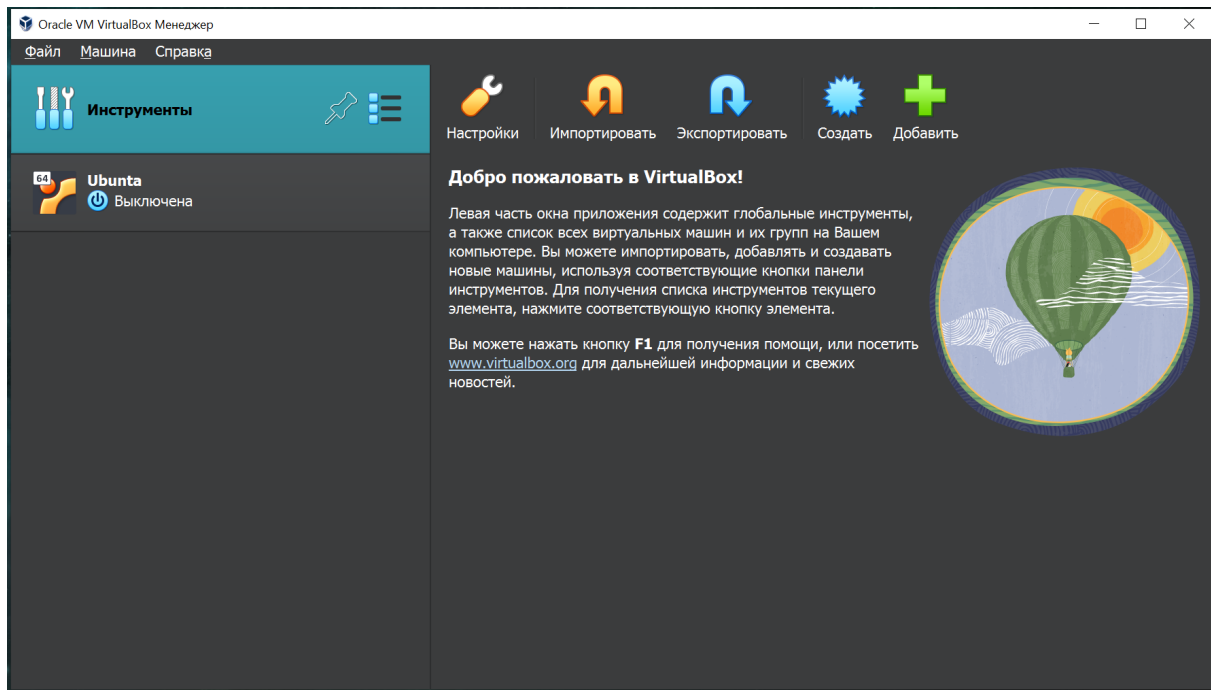
```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\Илья> wsl
ilia@DESKTOP-61RQEUN:/mnt/c/Users/Илья$
```

Установка VirtualBox и Ubuntu 24.04

VirtualBox - программный продукт виртуализации для операционных систем Windows, Linux, FreeBSD, macOS, Solaris/OpenSolaris, ReactOS, DOS и других. На виртуальную машину можно загрузить любую операционную систему используя файлы с расширением ".iso" соответствующей ОС.



Основные команды

Управление файлами и директориями

1. `ls` – список файлов и директорий
2. `cd` – смена директории
3. `pwd` – текущий путь
4. `mkdir` – создание директории
5. `rmdir` – удаление пустой директории
6. `rm` – удаление файла или директории:

```
rm file_name
```

```
rm -r directory_name
```

7. `cp` – копирование файлов или директорий:

```
cp source_file destination_file
```

```
cp -r source_directory destination_directory
```

8. **mv** – перемещение или переименование файлов или директорий:

```
mv old_name new_name
```

```
mv /path/to/file /new/path/to/file
```

9. **touch** – создание пустого файла или обновление времени последнего изменения:

```
touch file_name
```

10. **cat** – вывод содержимого файла:

```
cat file_name
```

11. **nano** или **vim** – текстовые редакторы для редактирования файлов:

```
nano file_name
```

```
vim file_name
```

Управление системой

1. **sudo** – выполнение команд от имени суперпользователя:

```
sudo command
```

2. **apt-get** или **apt** – пакетный менеджер для установки, обновления и удаления программ:

```
sudo apt-get update
```

 обновление списка пакетов

```
sudo apt-get upgrade
```

 обновление установленных пакетов

```
sudo apt-get install package_name
```

 установка пакета

```
sudo apt-get remove package_name
```

 удаление пакета

3. **ps** – просмотр списка запущенных процессов:

```
ps aux
```

4. `kill` – завершение процесса по PID:

```
kill PID
```

```
kill -9 PID
```

 принудительное завершение процесса

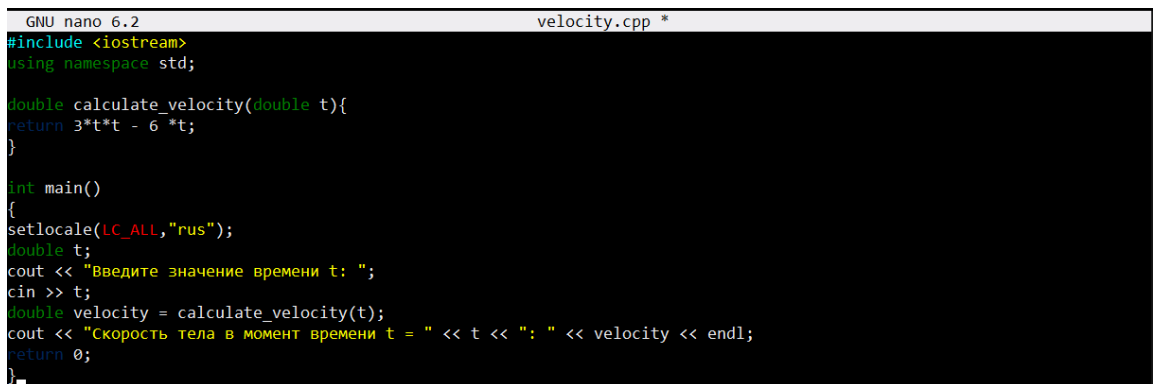
Создание простейшего приложения на языке C++ в ОС Ubuntu 24.04

Задача: Тело движется по закону $S = t^3 - 3t^2 + 2$. Вычислить скорость тела в момент времени t . Значение t ввести с клавиатуры (Функция скорости есть производная от функции расстояния по времени).

Шаги выполнения:

1. Создать файл `velocity.cpp`, зайти в редактор кода nano и написать текст программы:

```
nano velocity.cpp
```



```
GNU nano 6.2 velocity.cpp *
#include <iostream>
using namespace std;

double calculate_velocity(double t){
    return 3*t*t - 6*t;
}

int main()
{
    setlocale(LC_ALL, "rus");
    double t;
    cout << "Введите значение времени t: ";
    cin >> t;
    double velocity = calculate_velocity(t);
    cout << "Скорость тела в момент времени t = " << t << ": " << velocity << endl;
    return 0;
}
```

Для сохранения изменений нажать комбинацию клавиш Ctrl + O, затем Enter.

Для выхода из редактора nano нажать комбинацию Ctrl + X.

2. Установить компилятор "g++" командами:

```
sudo apt update
```

```
sudo apt install g++
```

3. Скомпилировать файл "velocity.cpp" командой:

```
g++ -o velocity velocity.cpp
```

4. Запустить программу командой:

```
./velocity
```

Результат выполнений шагов:

```
ilia@DESKTOP-61RQEUH:/mnt/c/Users/Илья/files$ ls
velocity.cpp
ilia@DESKTOP-61RQEUH:/mnt/c/Users/Илья/files$ g++ -o velocity velocity.cpp
ilia@DESKTOP-61RQEUH:/mnt/c/Users/Илья/files$ ls
velocity  velocity.cpp
ilia@DESKTOP-61RQEUH:/mnt/c/Users/Илья/files$ ./velocity
Введите значение времени t: 13
Скорость тела в момент времени t = 13: 429
ilia@DESKTOP-61RQEUH:/mnt/c/Users/Илья/files$
```

Создание простейшего приложения на языке C++ в ОС Windows

Задача: создать файл "hello.cpp", который выводит в консоль текст "Hello world!".

Шаги выполнения:

1. Создать файл hello.cpp
2. Найти в "Пуске" консоль "Developer Command Prompt for VS 2022". Эта консоль настроит необходимые переменные окружения для использования инструментов компиляции Visual Studio.
3. Перейти в директорию, в которой лежит файл "hello.cpp".
4. Ввести команду: `cl hello.cpp`
5. Запустить файл "hello.exe"

```
Developer Command Prompt for VS 2022
*****
** Visual Studio 2022 Developer Command Prompt v17.8.5
** Copyright (c) 2022 Microsoft Corporation
*****

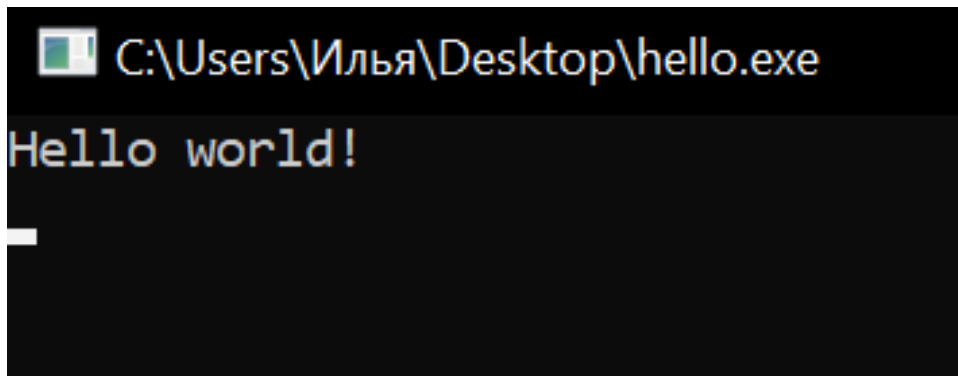
C:\Program Files\Microsoft Visual Studio\2022\Community> cd C:\Users\Илья\Desktop

C:\Users\Илья\Desktop>cl hello.cpp
Оптимизирующий компилятор Microsoft (R) C/C++ версии 19.38.33134 для x86
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

hello.cpp
C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.38.33130\include\ostream(779): warning C4530: И
спользован обработчик исключений C++, но семантика уничтожения объектов не включена. Задайте параметр /EHsc
C:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.38.33130\include\ostream(779): note: контекст с
оздания экземпляра шаблона (сначала самый старый)
hello.cpp(7): note: выполняется компиляция ссылки на экземпляр шаблон функции "std::basic_ostream<char,std::char_traits<
char>> &std::operator <<<std::char_traits<char>>(std::basic_ostream<char,std::char_traits<char>> &,const char *)"
Microsoft (R) Incremental Linker Version 14.38.33134.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:hello.exe
hello.obj
C:\Users\Илья\Desktop>
```

Запуск программы:



Управление кодом с помощью Git и документирование проекта

Я зарегистрировался на сайте GitHub и активировал курс Introduction to GitHub(<https://github.com/skills/introduction-to-github>):

Introduction to GitHub

Get started using GitHub in less than an hour.

Welcome


People use GitHub to build some of the most advanced technologies in the world. Whether you're visualizing data or building a new game, there's a whole community and set of tools on GitHub that can help you do it even better. GitHub Skills' "Introduction to GitHub" course guides you through everything you need to start contributing in less than an hour.

- **Who is this for:** New developers, new GitHub users, and students.
- **What you'll learn:** We'll introduce repositories, branches, commits, and pull requests.
- **What you'll build:** We'll make a short Markdown file you can use as your [profile README](#).
- **Prerequisites:** None. This course is a great introduction for your first day on GitHub.
- **How long:** This course takes less than one hour to complete.

In this course, you will:

1. Create a branch
2. Commit a file
3. Open a pull request
4. Merge your pull request

How to start this course

 [Start course](#)

Я прошёл этот курс. В нём рассматриваются: создание веток, коммиты, Pull Requests, создание репозитория, слияние веток.

Также мной пройден курс Communicate using Markdown. В нём рассматриваются: добавление headers, добавление изображений, добавление примера кода, создание списка задач, слияние pull requests.

Communicate using Markdown

Organize ideas and collaborate using Markdown, a lightweight language for text formatting.

Welcome

GitHub is about more than code. It's a platform for software collaboration, and Markdown is one of the most important ways developers can make their communication clear and organized in issues and pull requests. This course will walk you through creating and using headings more effectively, organizing thoughts in bulleted lists, and showing how much work you've completed with checklists. You can even use Markdown to add some depth to your work with the help of emoji, images, and links.

- **Who is this for:** New developers, new GitHub users, and students.
- **What you'll learn:** Use Markdown to add lists, images, and links in a comment or text file.
- **What you'll build:** We'll update a plain text file and add Markdown formatting, and you can use this file to start your own GitHub Pages site.
- **Prerequisites:** In this course you will work with pull requests as well as edit files. If these things aren't familiar to you, we recommend you take the [Introduction to GitHub](#) course, first!
- **How long:** This course takes less than one hour to complete.

In this course, you will:

1. Add headers
2. Add an image
3. Add a code example
4. Make a task list
5. Merge your pull request

Основные команды Git

`git --version` - версия

`git --config global user.name "username"` - установить глобальное имя в Гите

`git config user.name` - посмотреть глобальное имя в Гите

`git --config global user.email "mail"` - установить глобальную почту в Гите

`git config user.email` - посмотреть глобальную почту в Гите

`git init` - проинициализировать репозиторий (появится скрытая папка .git)

`git status` - посмотреть статус репозитория

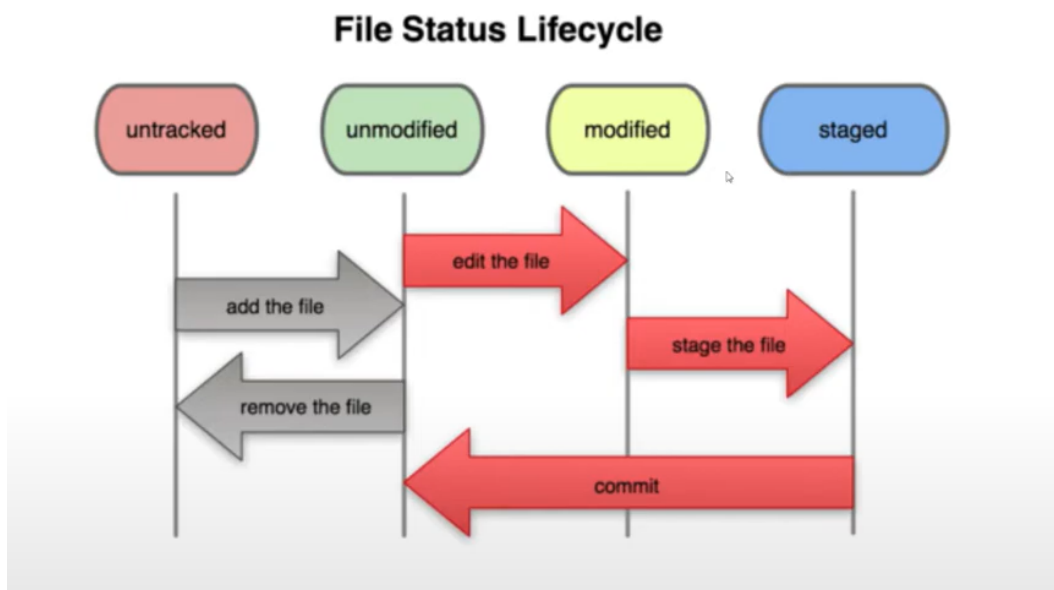
`git add main.cpp` - отслеживать файл main.cpp в репозитории

`git add *.cpp` - отслеживать ВСЕ файлы в папке с расширением .cpp

`git add .` - добавим ВСЕ не отслеживаемы файлы (важна точка) в наш локальный репозиторий

`git commit -m "added new file main.cpp"` - `-m` для того, чтобы добавить комментарий к коммиту.

Состояния файлов в git



Загрузка своего репозитория на github.

1. Создать репозиторий на github'e
2. Скопировать ссылку на репозиторий на githubе.
3. Зайти в репозиторий на ПК в гитбаше.
4. Прописать `git remote add origin https://github.com/smik91/testing_gitbush.git`
Так мы создали удалённый репозиторий под именем origin.

`git remote -v` - показать удаленные репозитории

`git push origin master` - загрузить все коммиты на гитхаб в ветку "master"

`git push -u origin master` - сделать гитпуш по умолчанию в master. (теперь можно писать просто `git push` и всё будет добавляться в "master" ветку)

Загрузка с github.

`git fetch origin` - добавляем в наш локальный репозиторий файлы с гитхаба.

`git merge origin/master` - вставляем файлы из локального репозитория в рабочую область

`git pull` - выполняет сразу 2 команды `git fetch origin` и `git merge origin/master`

Клонирование репозитория

Если нужно получить полную копию репозитория на githubе. (создается отдельная папка в месте назначения)

`git clone https://github.com/smik91/testing_gitbush`

Просмотр истории

`git log` - просмотр коммитов этого репозитория

`git log -p` - подробный просмотр изменений в каждом коммите

`git log --pretty=format:"%h %s"` - просмотр изменений в форме "%h %s"

`gitk` - графическое отображение коммитов и всего всего

Отмена изменений

`git commit --amend` - перезаписать прошлый коммит

`git reset HEAD <file>` - отменить добавление файла в staging area (сделать его не отслеживаемым)

`git checkout - <file>` - отмена изменений в рабочей директории внутри файла

Ветки

`git branch testing` - создать ветку "testing". (но указатель всё еще в мастер ветке)

`git checkout testing` - сделать указатель в ветку "testing"

`git checkout -b bugfix` - создать ветку "bugfix" и сразу переключиться на нее

`git merge testing` - слить ветку "testing" с той веткой на которой находимся

`git log --graph --oneline --all` - красивое графическое отображение коммитов

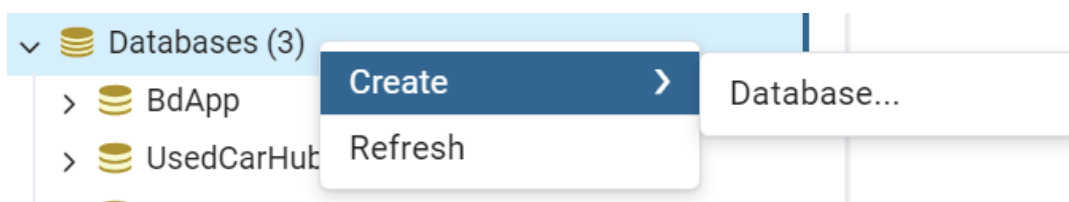
`git rebase master` - вставить ветку на которой мы находимся перед последним общим предком в ветку "master".

`git revert HEAD` - удаляет последний коммит на котором указатель HEAD/ либо можно указать конкретный хэш коммита. (создаёт новый коммит, который отменит действия удалённого коммита)

Работа с реляционной базой данных PostgreSQL.

Создание базы данных

1. Нажать кнопку "Create", а затем "Database"



2. Дать название базе данных и нажать кнопку "Save"

A screenshot of the 'Create - Database' dialog box in PostgreSQL. The 'General' tab is selected. The 'Database' field contains 'lab-bsu'. The 'OID' field is empty. The 'Owner' field is set to 'postgres'. The 'Comment' field is empty. At the bottom, there are buttons for 'Close', 'Reset', and 'Save'.

Задание

«Автомобиль»: id; марка; цвет; серийный номер; регистрационный номер; год выпуска; год техосмотра; цена.

Выполнить запросы:

- Вывести данные про автомобили, которым больше 5 лет.
- Создать таблицу vendor (id, ven_name, ven_description).
- Используя инструкцию alter, добавить дополнительные столбцы, один из которых vendor_id (тип integer и содержит идентификаторы заводов).

16

- Вывести данные обо всех автомобилях в форме идентификатор автомобиля, наименование, год выпуска, название завода.
- подсчет количества машин с помощью count, если стоимость > 16000 руб
- суммарная стоимость машин с помощью sum, если год выпуска = 2016
- максимальная и минимальная стоимость с помощью max и min
- Используя инструкцию inner join вывести полные сведения о машинах и заводе для завода с id=3.

Создание таблицы

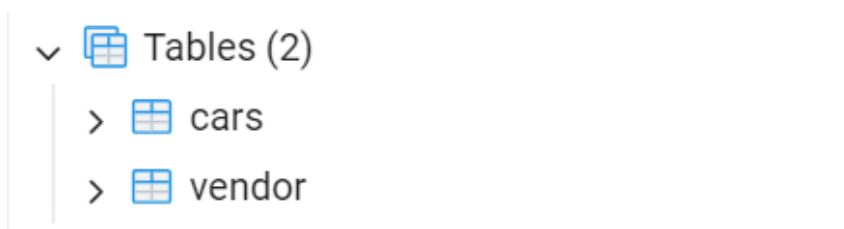
Для создания таблицы нужно выполнить следующий SQL запрос:

Query	Query History
1	CREATE TABLE public.Cars (
2	id SERIAL PRIMARY KEY,
3	brand VARCHAR(100) NOT NULL,
4	serial_number VARCHAR(100) UNIQUE NOT NULL,
5	registration_number VARCHAR(100) UNIQUE NOT NULL,
6	year_of_manufacture INT NOT NULL,
7	year_of_inspection INT,
8	price NUMERIC(10, 2)
9);

Для создания таблицы vendor (производители) нужно выполнить следующий SQL запрос:

Query	Query History
1	CREATE TABLE public.vendor (
2	id SERIAL PRIMARY KEY,
3	ven_name VARCHAR(100) NOT NULL,
4	ven_description VARCHAR(100) NOT NULL
5);

В результате появятся 2 таблицы "cars" и "vendor"



Добавление данных в таблицы

Для добавления данных в таблицу "cars" требуется выполнить следующий SQL запрос

Query	Query History
1	INSERT INTO public.Cars (brand, serial_number, registration_number, year_of_manufacture, year_of_inspection, price)
2	VALUES
3	('Toyota', 'SN123456', 'REG1234', 2020, 2022, 15000.00),
4	('Honda', 'SN234567', 'REG2345', 2018, 2020, 12000.00),
5	('Ford', 'SN345678', 'REG3456', 2019, 2021, 18000.00),
6	('BMW', 'SN456789', 'REG4567', 2017, 2019, 25000.00),
7	('Audi', 'SN567890', 'REG5678', 2021, 2023, 30000.00),
8	('Mercedes', 'SN678901', 'REG6789', 2016, 2018, 35000.00),
9	('Volkswagen', 'SN789012', 'REG7890', 2015, 2017, 14000.00),
10	('Hyundai', 'SN890123', 'REG8901', 2022, 2024, 16000.00),
11	('Nissan', 'SN901234', 'REG9012', 2023, 2025, 20000.00),
12	('Chevrolet', 'SN012345', 'REG0123', 2014, 2016, 22000.00);

Data Output	Messages	Notifications
INSERT 0 10		
Query returned successfully in 46 msec.		

Проверим, что все данные успешно добавлены

Query

Query History

Scratch Pad

X

1

SELECT * FROM public.cars

2

ORDER BY id ASC

Data Output

Messages

Notifications

id

[PK] integer

brand

character varying (100)

serial_number

character varying (100)

registration_number

character varying (100)

year_of_manufacture

integer

year_of_inspection

integer

price

numeric (10,2)

1

1

Toyota

SN123456

REG1234

2020

2022

15000.00

2

2

Honda

SN234567

REG2345

2018

2020

12000.00

3

3

Ford

SN345678

REG3456

2019

2021

18000.00

4

4

BMW

SN456789

REG4567

2017

2019

25000.00

5

5

Audi

SN567890

REG5678

2021

2023

30000.00

6

6

Mercedes

SN678901

REG6789

2016

2018

35000.00

7

7

Volkswagen

SN789012

REG7890

2015

2017

14000.00

8

8

Hyundai

SN890123

REG8901

2022

2024

16000.00

9

9

Nissan

SN901234

REG9012

2023

2025

20000.00

10

10

Chevrolet

SN012345

REG0123

2014

2016

22000.00

Для добавления данных в таблицу "vendor" требуется выполнить следующий SQL запрос

Query Query History	
<pre>1 INSERT INTO public.vendor (ven_name, ven_description) 2 VALUES 3 ('AutoWorld', 'Leading automobile retailer specializing in a wide range of cars.'), 4 ('CarMax', 'Nationwide chain offering a large selection of used cars with no-haggle pricing.'), 5 ('MotorCity', 'Local car dealer known for excellent customer service and extensive inventory.'), 6 ('DriveTime', 'Dealer offering easy financing options and a diverse selection of vehicles.'), 7 ('AutoNation', 'Largest automotive retailer in the US, providing a vast selection of cars.'), 8 ('Carvana', 'Online used car dealer known for its car vending machines and home delivery.');</pre>	
Data Output Messages Notifications	
INSERT 0 6	
Query returned successfully in 26 msec.	

Проверим, что все данные успешно добавлены

Query

Query History

1

SELECT * FROM public.vendor

2

ORDER BY id ASC

Data Output

Messages

Notifications

	id	ven_name	ven_description
	[PK] integer	character varying (100)	character varying (100)
1	1	AutoWorld	Leading automobile retailer specializing in a wide range of cars.
2	2	CarMax	Nationwide chain offering a large selection of used cars with no-haggle prici...
3	3	MotorCity	Local car dealer known for excellent customer service and extensive invento...
4	4	DriveTime	Dealer offering easy financing options and a diverse selection of vehicles.
5	5	AutoNation	Largest automotive retailer in the US, providing a vast selection of cars.
6	6	Carvana	Online used car dealer known for its car vending machines and home delivery.

Выполнение заданий

1. Вывести данные про автомобили, которым больше 5 лет.

Query		Query History						
1		SELECT * FROM public.cars						
2		WHERE EXTRACT(YEAR FROM CURRENT_DATE) - year_of_manufacture > 5;						
Data Output		Messages						
	id [PK] integer	brand character varying (100)	serial_number character varying (100)	registration_number character varying (100)	year_of_manufacture integer	year_of_inspection integer	price numeric (10,2)	
1	2	Honda	SN234567	REG2345	2018	2020	12000.00	
2	4	BMW	SN456789	REG4567	2017	2019	25000.00	
3	6	Mercedes	SN678901	REG6789	2016	2018	35000.00	
4	7	Volkswagen	SN789012	REG7890	2015	2017	14000.00	
5	10	Chevrolet	SN012345	REG0123	2014	2016	22000.00	

2. Используя инструкцию alter, добавить дополнительные столбцы, один из которых vendor_id (тип integer и содержит идентификаторы заводов)

Query		Query History						
1		ALTER TABLE public.Cars						
2		ADD COLUMN vendor_id INTEGER;						
Data Output		Messages						
		ALTER TABLE						
		Query returned successfully in 52 msec.						

Результат

Query		Query History							Scratch Pad X
1		SELECT * FROM public.cars							
2		ORDER BY id ASC							
Data Output		Messages							
	id	brand character varying (100)	serial_number character varying (100)	registration_number character varying (100)	year_of_manufacture integer	year_of_inspection integer	price numeric (10,2)	vendor_id integer	
1	1	Toyota	SN123456	REG1234	2020	2022	15000.00	[null]	
2	2	Honda	SN234567	REG2345	2018	2020	12000.00	[null]	
3	3	Ford	SN345678	REG3456	2019	2021	18000.00	[null]	
4	4	BMW	SN456789	REG4567	2017	2019	25000.00	[null]	
5	5	Audi	SN567890	REG5678	2021	2023	30000.00	[null]	
6	6	Mercedes	SN678901	REG6789	2016	2018	35000.00	[null]	
7	7	Volkswagen	SN789012	REG7890	2015	2017	14000.00	[null]	
8	8	Hyundai	SN890123	REG8901	2022	2024	16000.00	[null]	
9	9	Nissan	SN901234	REG9012	2023	2025	20000.00	[null]	
10	10	Chevrolet	SN012345	REG0123	2014	2016	22000.00	[null]	

3. Вывести данные обо всех автомобилях в форме: идентификатор автомобиля,наименование, год выпуска, название завода.

Для этого заполним колонку "vendor_id" в таблице "cars"

Query

Query History

```
1 UPDATE public.Cars
2 SET vendor_id = CASE
3     WHEN id = 1 THEN 1
4     WHEN id = 2 THEN 2
5     WHEN id = 3 THEN 3
6     WHEN id = 4 THEN 4
7     WHEN id = 5 THEN 5
8     WHEN id = 6 THEN 6
9     WHEN id = 7 THEN 1
10    WHEN id = 8 THEN 2
11    WHEN id = 9 THEN 3
12    WHEN id = 10 THEN 4
13 END;
```

Data Output

Messages

Notifications

UPDATE 10

Query returned successfully in 52 msec.

В результате колонка "vendor_id" заполнится следующим образом

	vendor_id integer
1	1
2	2
3	3
4	4
5	5
6	6
7	1
8	2
9	3
10	4

Выполняем SQL запрос для получения таблицы в нужном нам виде

Query

Query History

1

SELECT

2

c.id,

3

c.brand,

4

c.year_of_manufacture,

5

v.ven_name

6

FROM

7

public.Cars c

8

INNER JOIN

9

public.Vendor v ON c.vendor_id = v.id;

Data Output

Messages

Notifications

	id integer	brand character varying (100)	year_of_manufacture integer	ven_name character varying (100)
1	1	Toyota	2020	AutoWorld
2	2	Honda	2018	CarMax
3	3	Ford	2019	MotorCity
4	4	BMW	2017	DriveTime
5	5	Audi	2021	AutoNation
6	6	Mercedes	2016	Carvana
7	7	Volkswagen	2015	AutoWorld
8	8	Hyundai	2022	CarMax
9	9	Nissan	2023	MotorCity
10	10	Chevrolet	2014	DriveTime

4. Подсчет количества машин с помощью count, если стоимость > 16000 руб

Query

Query History

1

SELECT

2

COUNT(*) AS count_of_cars

3

FROM

4

public.Cars

5

WHERE

6

price > 16000;

Data Output

Messages

Notifications

</

5. Суммарная стоимость машин с помощью sum, если год выпуска 2016

Query

Query History

1

2

3

4

5

6

SELECT

SUM(price) AS total_cost

FROM

public.Cars

WHERE

year_of_manufacture = 2016;

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	total_cost numeric 🔒
1	35000.00

6. Максимальная и минимальная стоимость машин с помощью "sum" и "min"

Query

Query History

1

2

3

4

5

SELECT

MAX(price) AS max_price,

MIN(price) AS min_price

FROM

public.Cars;

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	max_price numeric 🔒	min_price numeric 🔒
1	35000.00	12000.00

7. Используя инструкцию inner join вывести полные сведения о машинах и заводе

для завода с id=3.

```
Query  Query History
1  SELECT
2      c.*,
3      v.*
4  FROM
5      public.Cars c
6  INNER JOIN
7      public.Vendor v ON c.vendor_id = v.id
8  WHERE
9      v.id = 3;
```

Результат

Data Output Messages Notifications												
	id	brand	serial_number	registration_number	year_of_manufacture	year_of_inspection	price	vendor_id	id	ven_name	ven_description	
1	integer	character varying (100)	character varying (100)	character varying (100)	integer	integer	numeric (10,2)	integer	integer	character varying (100)	character varying (100)	
2	3	Ford	SN345678	REG3456	2019	2021	18000.00	3	3	MotorCity	Local car dealer known fo	
3	9	Nissan	SN901234	REG9012	2023	2025	20000.00	3	3	MotorCity	Local car dealer known fo	

Источники

1. Б. Керниган, Д. Ритчи. Язык программирования C. Второе издание. — М.: "Вильямс", 2008. — 304 с.
2. Х. Дейтел, П. Дейтел. Как программировать на C.
3. Н.А. Калинина, Н.И. Костюкова. Основы программирования на языке C. (<http://www.intuit.ru/studies/courses/43/43/info>)
4. Брайен В., Деннис М. Язык программирования C (<http://www.intuit.ru/studies/courses/97/97/info>)
5. Н.Н. Иванов. Программирование в Linux. Самоучитель. — СПб.: БХВПетербург, 2007.- 416 с.
6. Начинаящим о работе в терминале — <http://appleinsider.ru/tips-tricks/mac-os-xnachinayushhim-o-rabote-v-terminale.html>

7. Начинаем работать с VMware Workstation (для чайников) —

http://all-ht.ru/inf/vpc/p_0_2.html

8. Редакторы nano и vim — <https://www.youtube.com/watch?v=R33F0EDivwk>

9. 56 полезных команд терминала в macOS на все случаи жизни —

<https://lifel hacker.ru/komandy-terminala-macos/>

10. Консоль для маководов: Beyond the GUI —

<https://habr.com/ru/articles/143341/>

10. Основы работы с командной строкой (Терминалом) на Mac OS X —

<https://vertex-academy.com/tutorials/ru/komandy-komandnaya-stroka-terminal-vmacos/>

11. Mac Terminal cheatsheet —

https://zhuosongz.github.io/docs/cheatsheets/mac_terminal.pdf

12. Github Git Guide - <https://github.com/git-guides>

13. Руководство по PostgreSQL metanit.com -
(<https://metanit.com/sql/postgresql>)