

Automatically Generate a PDF and send it by Email

1 hour 30 minutesFree
Rate Lab

Introduction

You work for a company that sells second hand cars. Management wants to get a summary of the amounts of vehicles that have been sold at the end of every month. The company already has a web service which serves sales data at the end of every month but management wants an email to be sent out with an attached PDF so that data is more easily readable.

What you'll do

- Write a script that summarizes and processes sales data into different categories
- Generate a PDF using Python
- Automatically send a PDF by email

You'll have 90 minutes to complete this lab.

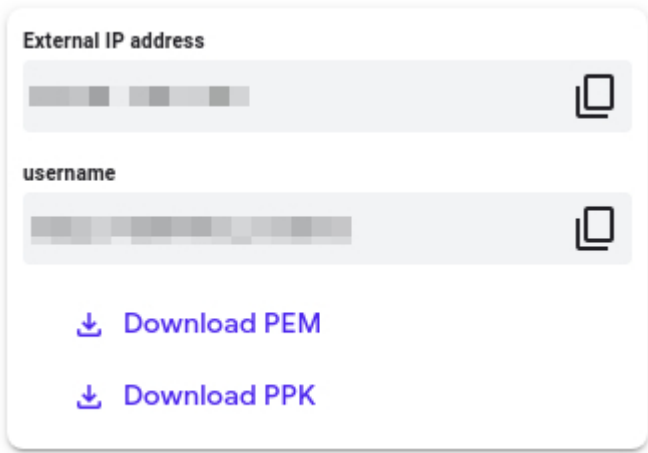
Start the lab

You'll need to start the lab before you can access the materials in the virtual machine OS. To do this, click the green “Start Lab” button at the top of the screen.

Note: For this lab you are going to access the **Linux VM** through your **local SSH Client**, and not use the **Google Console** (**Open GCP Console** button is not available for this lab).

A green rectangular button with the text "Start Lab" in white.

After you click the “Start Lab” button, you will see all the SSH connection details on the left-hand side of your screen. You should have a screen that looks like this:

A light gray rounded rectangle containing SSH connection details. It has two input fields: "External IP address" and "username", each with a copy icon to its right. Below these are two download links: "Download PEM" and "Download PPK", each preceded by a download icon.

External IP address

username

↓ Download PEM

↓ Download PPK

Accessing the virtual machine

Please find one of the three relevant options below based on your device's operating system.

Note: Working with Qwiklabs may be similar to the work you'd perform as an **IT Support Specialist**; you'll be interfacing with a cutting-edge technology that requires multiple steps to access, and perhaps healthy doses of patience and persistence(!). You'll also be using **SSH** to enter the labs -- a critical skill in IT Support that you'll be able to practice through the labs.

Option 1: Windows Users: Connecting to your VM

In this section, you will use the PuTTY Secure Shell (SSH) client and your VM's External IP address to connect.

Download your PPK key file

You can download the VM's private key file in the PuTTY-compatible **PPK** format from the Qwiklabs Start Lab page. Click on **Download PPK**.

 [Download PEM](#)

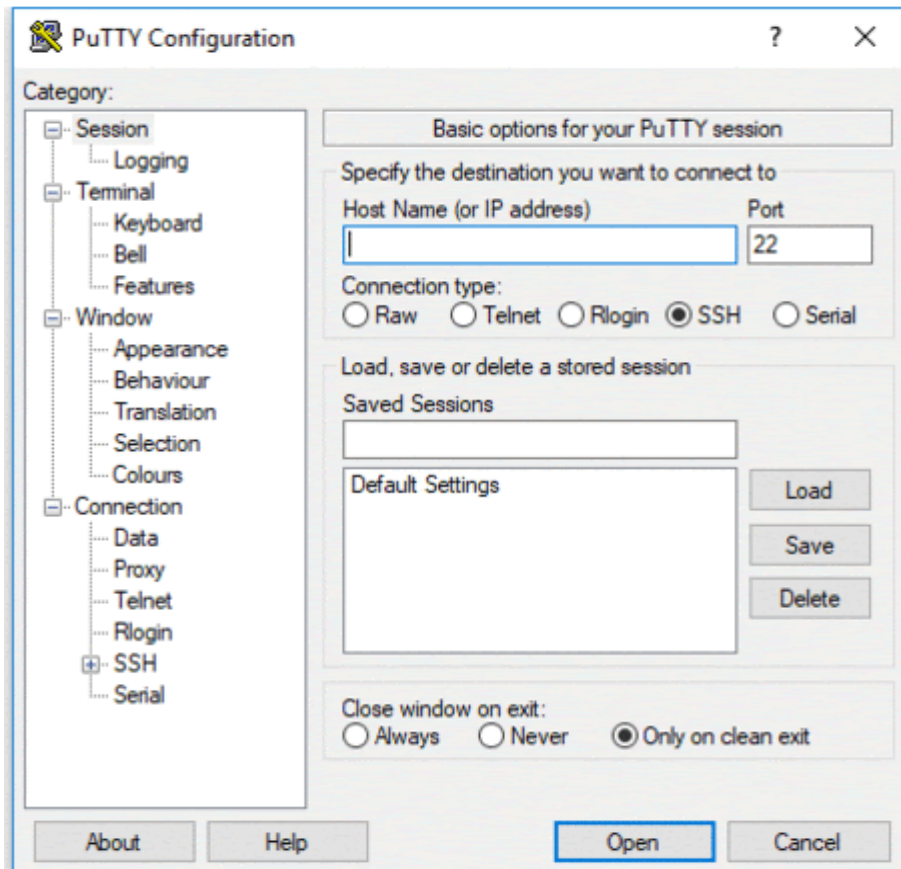
 [Download PPK](#)



Connect to your VM using SSH and PuTTY

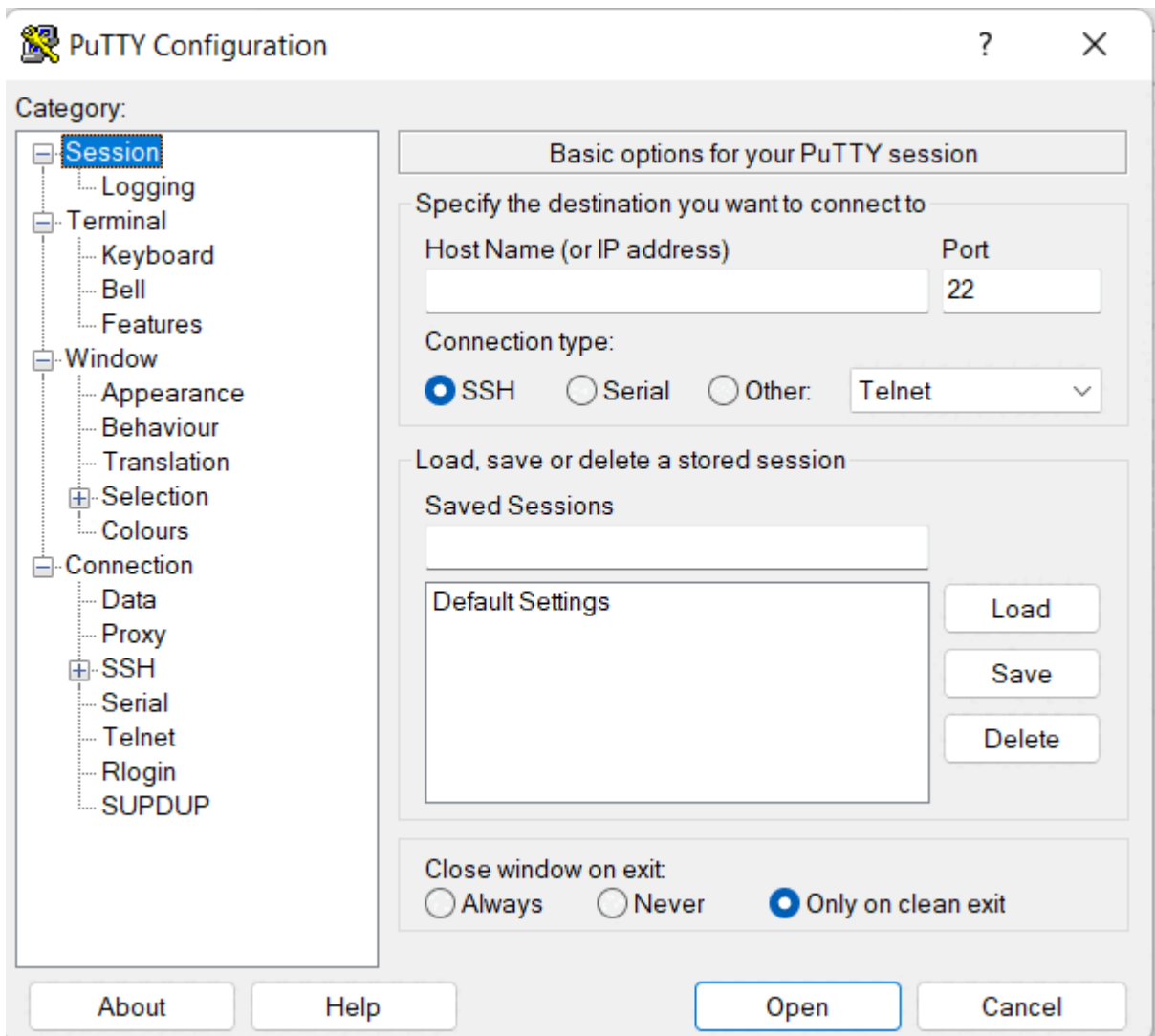
1. You can download Putty from [here](#)
2. In the **Host Name (or IP address)** box, enter `username@external_ip_address`.

Note: Replace **username** and **external_ip_address** with values provided in the lab.



3. In the **Connection** list, expand **SSH**.
4. Then expand **Auth** by clicking on + icon.
5. Now, select the **Credentials** from the **Auth** list.
6. In the **Private key file for authentication** box, browse to the PPK file that you downloaded and double-click it.
7. Click on the **Open** button.

Note: PPK file is to be imported into PuTTY tool using the Browse option available in it. It should not be opened directly but only to be used in PuTTY.



8. Click **Yes** when prompted to allow a first connection to this remote SSH server. Because you are using a key pair for authentication, you will not be prompted for a password.

Common issues

If PuTTY fails to connect to your Linux VM, verify that:

- You entered **<username>@<external ip address>** in PuTTY.
- You downloaded the fresh new PPK file for this lab from Qwiklabs.
- You are using the downloaded PPK file in PuTTY.

Option 2: OSX and Linux users: Connecting to your VM via SSH

Download your VM's private key file.

You can download the private key file in PEM format from the Qwiklabs Start Lab page. Click on **Download PEM**.



Connect to the VM using the local Terminal application

A **terminal** is a program which provides a **text-based interface for typing commands**. Here you will use your terminal as an SSH client to connect with lab provided Linux VM.

1. Open the Terminal application.
 - To open the terminal in Linux use the shortcut key **Ctrl+Alt+t**.
 - To open terminal in **Mac (OSX)** enter **cmd + space** and search for **terminal**.
2. Enter the following commands.

Note: Substitute the **path/filename for the PEM** file you downloaded, **username** and **External IP Address**.

You will most likely find the PEM file in **Downloads**. If you have not changed the download settings of your system, then the path of the PEM key will be **~/Downloads/qwikLABS-XXXXX.pem**

```
chmod 600 ~/Downloads/qwikLABS-XXXXX.pem
```

Copied!

content_copy

```
ssh -i ~/Downloads/qwikLABS-XXXXX.pem username@External Ip Address
```

Copied!

content_copy

```
~$ ssh -i ~/Downloads/qwikLABS-L923-42090.pem gcpstagingedit1370_stu
The authenticity of host '35.239.106.192 (35.239.106.192)' can't be established.
ECDSA key fingerprint is SHA256:vrz8b4aYUtruFh0A6wZn6Ozy1oqqPEfh931olvxITm8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '35.239.106.192' (ECDSA) to the list of known hosts.
Linux linux-instance 4.9.0-9-amd64 #1 SMP Debian 4.9.168-1+deb9u2 (2019-05-13) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
gcpstagingedit1370_student@linux-instance:~$
```

Option 3: Chrome OS users: Connecting to your VM via SSH

Note: Make sure you are not in **Incognito/Private mode** while launching the application.

Download your VM's private key file.

You can download the private key file in PEM format from the Qwiklabs Start Lab page. Click on **Download PEM**.



Connect to your VM

1. Add Secure Shell from here to your Chrome browser.
2. Open the Secure Shell app and click on **[New Connection]**.

[New Connection]

username@hostname or free form

username

hostname

SSH relay server options

Identity: [default]

SSH Arguments: extra command line

Current profile: default

Mount Path: the default path

[DEL] Delete

Options

3. In the **username** section, enter the username given in the Connection Details Panel of the lab. And for the **hostname** section, enter the external IP of your VM instance that is mentioned in the Connection Details Panel of the lab.

[New Connection]

username@hostname or free form text

username

hostname

SSH relay server options

Identity: [default]

SSH Arguments: extra command line arg

Current profile: default

Mount Path: the default path is th

[DEL] Delete

Options

4. In the **Identity** section, import the downloaded PEM key by clicking on the **Import...** button beside the field. Choose your PEM key and click on the **OPEN** button.

Note: If the key is still not available after importing it, refresh the application, and select it from the **Identity** drop-down menu.

5. Once your key is uploaded, click on the **[ENTER] Connect** button below.

[New Connection]

username@hostname or free form text

username

hostname

SSH relay server options

Identity: [default]

SSH Arguments: extra command line arguments

Current profile: default

Mount Path: the default path is the

[DEL] Delete

Options

6. For any prompts, type **yes** to continue.

7. You have now successfully connected to your Linux VM.

You're now ready to continue with the lab!

Sample report

In this section, you will be creating a PDF report named "**A Complete Inventory of My Fruit**". The script to generate this report and send it by email is already pre-done. You can have a look at the script in the `scripts` directory.

```
ls ~/scripts  
Copied!  
content_copy
```

Output:

```
student-02-28a8a5b666b8@linux-instance:~$ ls ~/scripts  
cars.py  emails.py  example.py  reports.py
```

In the `scripts` directory, you will find `reports.py` and `emails.py` files. These files are used to **generate PDF files** and **send emails** respectively.

Take a look at these files using `cat` command.

```
cat ~/scripts/reports.py  
Copied!  
content_copy
```

Output:

```
student-02-28a8a5b666b8@linux-instance:~$ cat ~/scripts/reports.py
#!/usr/bin/env python3

from reportlab.platypus import SimpleDocTemplate
from reportlab.platypus import Paragraph, Spacer, Table, Image
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.lib import colors

def generate(filename, title, additional_info, table_data):
    styles = getSampleStyleSheet()
    report = SimpleDocTemplate(filename)
    report_title = Paragraph(title, styles["h1"])
    report_info = Paragraph(additional_info, styles["BodyText"])
    table_style = [('GRID', (0,0), (-1,-1), 1, colors.black),
                   ('FONTNAME', (0,0), (-1,0), 'Helvetica-Bold'),
                   ('ALIGN', (0,0), (-1,-1), 'CENTER')]
    report_table = Table(data=table_data, style=table_style, hAlign="LEFT")
    empty_line = Spacer(1,20)
    report.build([report_title, empty_line, report_info, empty_line, report_table])
```

cat ~/scripts/emails.py

Copied!

content_copy

Output:

```

student-02-28a8a5b666b8@linux-instance:~$ cat ~/scripts/emails.py
#!/usr/bin/env python3

import email.message
import mimetypes
import os.path
import smtplib

def generate(sender, recipient, subject, body, attachment_path):
    """Creates an email with an attachment."""
    # Basic Email formatting
    message = email.message.EmailMessage()
    message["From"] = sender
    message["To"] = recipient
    message["Subject"] = subject
    message.set_content(body)

    # Process the attachment and add it to the email
    attachment_filename = os.path.basename(attachment_path)
    mime_type, _ = mimetypes.guess_type(attachment_path)
    mime_type, mime_subtype = mime_type.split('/', 1)

    with open(attachment_path, 'rb') as ap:
        message.add_attachment(ap.read(),
                               maintype=mime_type,
                               subtype=mime_subtype,
                               filename=attachment_filename)

    return message

def send(message):
    """Sends the message to the configured SMTP server."""
    mail_server = smtplib.SMTP('localhost')
    mail_server.send_message(message)
    mail_server.quit()

```

Now, take a look at `example.py`, which uses these two modules **reports** and **emails** to create a report and then send it by email.

```

cat ~/scripts/example.py
Copied!
content_copy

```

Grant executable permission to the `example.py` script.

```

sudo chmod o+wx ~/scripts/example.py
Copied!
content_copy

```

Run the `example.py` script, which will generate mail to you.

```
./scripts/example.py
```

Copied!

```
content_copy
```

A mail should now be successfully sent.

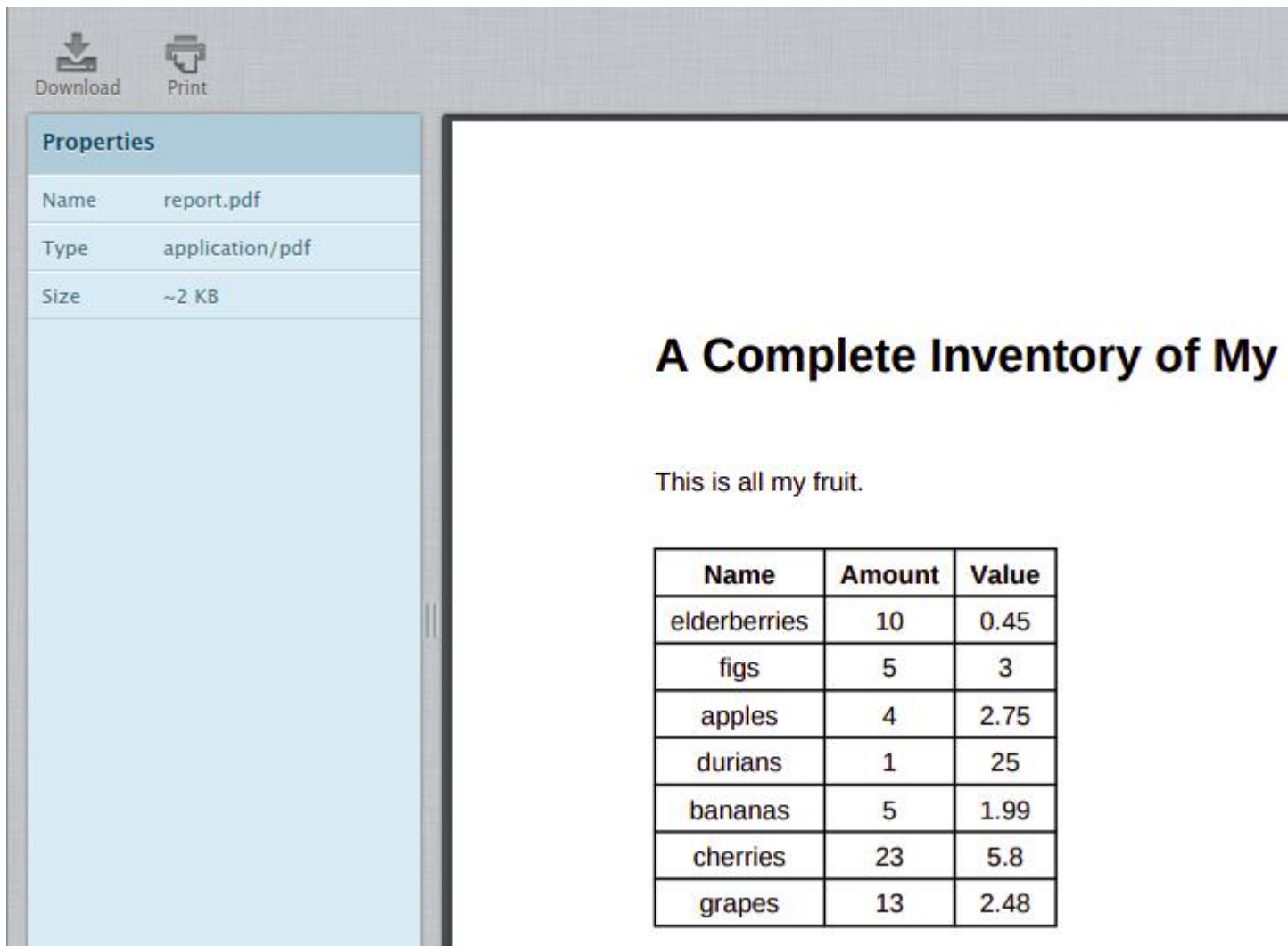
Copy the `external IP address` of your instance from the Connection Details Panel on the left side and open a new web browser tab and enter the IP address. The Roundcube Webmail login page appears.

Here, you'll need a login to **roundcube** using the username and password mentioned in the Connection Details Panel on the left hand side, followed by clicking **Login**.



Now you should be able to see your inbox, with one unread email. Open the mail by double clicking on it. There should be a report in PDF format attached to the mail. View the report by opening it.

Output:



Generate report

Now, let's make a couple of changes in the `example.py` file to add a new fruit and change the sender followed by granting editor permission. Open `example.py` file using the following command:

```
nano ~/scripts/example.py
Copied!
content_copy
```

And update the following variables:

variable_name	value
sender	Replace sender@example.com with automation@example.co

	m
table_data	Add another entry into the list: <code>['kiwi', 4, 0.49]</code>

The file should now look similar to:

```
#!/usr/bin/env python3
import emails
import os
import reports
table_data=[
    ['Name', 'Amount', 'Value'],
    ['elderberries', 10, 0.45],
    ['figs', 5, 3],
    ['apples', 4, 2.75],
    ['durians', 1, 25],
    ['bananas', 5, 1.99],
    ['cherries', 23, 5.80],
    ['grapes', 13, 2.48],
    ['kiwi', 4, 0.49]]
reports.generate("/tmp/report.pdf", "A Complete Inventory of My Fruit", "This is all my fruit.", table_data)
sender = "automation@example.com"
receiver = "{}@example.com".format(os.environ.get('USER'))
subject = "List of Fruits"
body = "Hi\n\nI'm sending an attachment with all my fruit."
message = emails.generate(sender, receiver, subject, body, "/tmp/report.pdf")
emails.send(message)
Copied!
content_copy
```

Once you've made the changes in the `example.py` script, save the file by typing **Ctrl-o**, **Enter** key and **Ctrl-x**.

Now execute the example script again.

```
./scripts/example.py
Copied!
content_copy
```

Now, check the webmail for any new mail. You can click on the **Refresh** button to refresh your inbox.

Properties	
Name	report.pdf
Type	application/pdf
Size	~2 KB

A Complete Inventory

This is all my fruit.

Name	Amount	Value
elderberries	10	0.45
figs	5	3
apples	4	2.75
durians	1	25
bananas	5	1.99
cherries	23	5.8
grapes	13	2.48
kiwi	4	0.49

Click Check my progress to verify the objective.

Generate sample report

Check my progress

Sales summary

In this section, let's view the summary of last month's sales for all the models offered by the company. This data is in a JSON file named `car_sales.json`. Let's have a look at it.

```
cat car_sales.json
```

Copied!

```
content_copy
```

Output:

```
student-02-28a8a5b666b8@linux-instance:~$ cat car_sales.json
[{"id":1,"car":{"car_make":"Ford","car_model":"Club Wagon","car_year":1997},"pri
{"id":2,"car":{"car_make":"Acura","car_model":"TL","car_year":2005},"price":"$14
{"id":3,"car":{"car_make":"Volkswagen","car_model":"Jetta","car_year":2009},"pri
{"id":4,"car":{"car_make":"Chevrolet","car_model":"Uplander","car_year":2006},"p
{"id":5,"car":{"car_make":"Plymouth","car_model":"Roadrunner","car_year":1969},"
{"id":6,"car":{"car_make":"GMC","car_model":"Safari","car_year":2000},"price":"$
{"id":7,"car":{"car_make":"Lamborghini","car_model":"Murciélago","car_year":2003
{"id":8,"car":{"car_make":"GMC","car_model":"3500","car_year":1999},"price":"$19
{"id":9,"car":{"car_make":"Maybach","car_model":"62","car_year":2004},"price":"$
{"id":10,"car":{"car_make":"Chevrolet","car_model":"Cavalier","car_year":2001},"
```

To simplify the JSON structure, here is an example of one of the JSON objects among the list.

```
{
  "id": 47,
  "car": {
    "car_make": "Lamborghini",
    "car_model": "Murciélago",
    "car_year": 2002
  },
  "price": "$13724.05",
  "total_sales": 149
}
```

Copied!

```
content_copy
```

Here `id`, `car`, `price` and `total_sales` are the field names (key).

The script `cars.py` already contains part of the work, but learners need to complete the task by writing the remaining pieces. The script already calculates the car model with the most revenue (`price * total_sales`) in the `process_data` method. Learners need to add the following:

1. Calculate the car model which had the most sales by completing the `process_data` method, and then appending a formatted string to the `summary` list in the below format:

- "The {car model} had the most sales: {total sales}"

2. Calculate the most popular car_year across all car make/models (in other words, find the total count of cars with the car_year equal to 2005, equal to 2006, etc. and then figure out the most popular year) by completing the `process_data` method, and append a formatted string to the `summary` list in the below format:

- "The most popular year was {year} with {total sales in that year} sales."

The challenge

Here, you are going to update the script `cars.py`. You will be using the above JSON data to process information. A part of the script is already done for you, where it calculates the car model with the most revenue (price * total_sales). You should now fulfil the following objectives with the script:

1. Calculate the car model which had the most sales.
 - a. Call `format_car` method for the car model.
2. Calculate the most popular car_year across all car make/models.

Hint: Find the total count of cars with the car_year equal to 2005, equal to 2006, etc. and then figure out the most popular year.

Grant required permissions to the file `cars.py` and open it using nano editor.

```
sudo chmod o+wx ~/scripts/cars.py
```

Copied!

content_copy

```
nano ~/scripts/cars.py
```

Copied!

content_copy

The code is well commented including the TODO sections for you to understand and fulfill the objectives.

Generate PDF and send Email

Once the data is collected, you will also need to further update the script to generate a PDF report and automatically send it through email.

To generate a PDF:

- Use the `reports.generate()` function within the main function.
- The report should be named as **cars.pdf**, and placed in the folder **/tmp/**.
- The PDF should contain:
 1. A summary paragraph which contains the most sales/most revenue/most popular year values worked out in the previous step.
Note: To add line breaks in the PDF, use: `
` between the lines.
 2. A table which contains all the information parsed from the JSON file, organised by `id_number`. The car details should be combined into one column in the form `<car_make>` `<car_model>` (`<car_year>`).

Note: You can use the `cars_dict_to_table` function for the above task.

Example:

ID	Car	Price	Total Sales
47	Acura TL (2007)	€14459,15	1192
73	Porsche 911 (2010)	€6057,74	882
85	Mercury Sable (2005)	€45660,46	874

To send the PDF through email:

Once the PDF is generated, you need to send the email, using the `emails.generate()` and `emails.send()` methods.

Use the following details to pass the parameters to `emails.generate()`:

- **From:** automation@example.com
- **To:** <user>@example.com
- **Subject line:** Sales summary for last month

- **E-mail Body:** The same summary from the PDF, but using `\n` between the lines
- **Attachment:** Attach the PDF path i.e. generated in the previous step

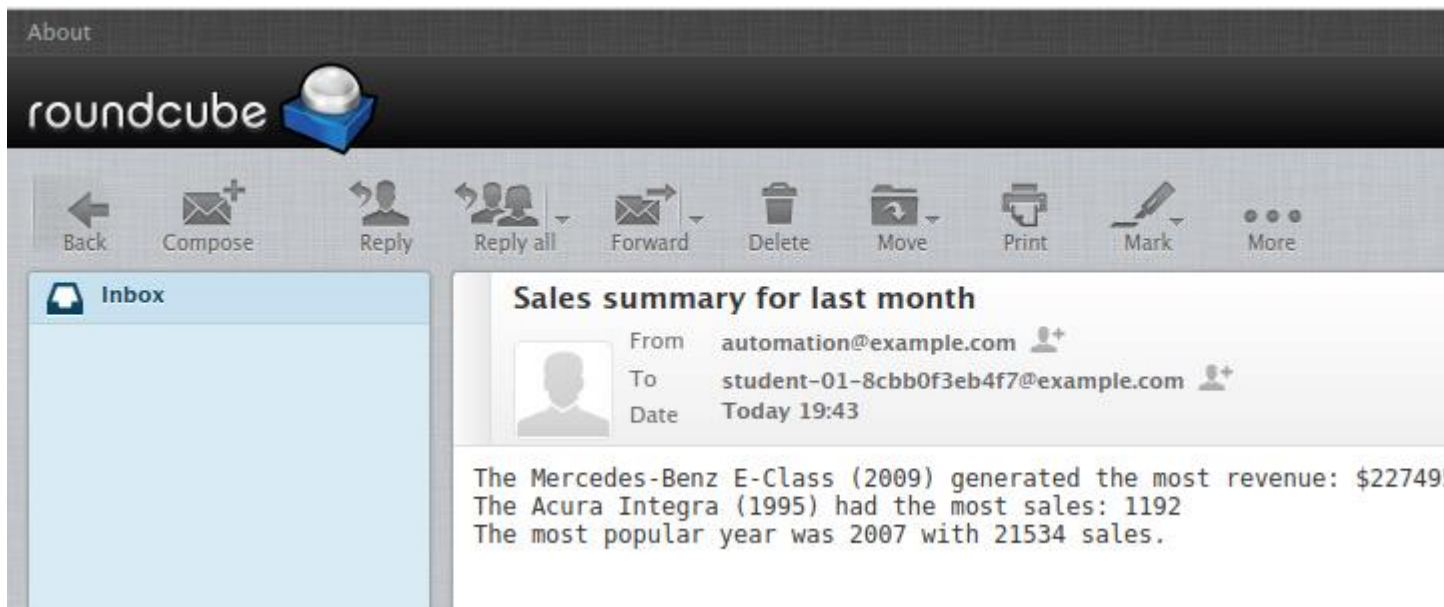
Once you have completed editing `cars.py` script, save the file by typing **Ctrl-o**, **Enter** key, and **Ctrl-x**.

Run the `cars.py` script, which will generate mail to their user.


```
./scripts/cars.py
Copied!
content_copy
```


Now, check the webmail for any new mail. You can click on the **Refresh** button to refresh your inbox.

Output:



Open `cars.pdf` that's located on the right most side.


Download


Print

Properties

Namecars.pdf

Typeapplication/pdf

Size~37 KB

Sales summary for

The Mercedes-Benz E-Class (2009)
The Acura Integra (1995) had the m
The most popular year was 2007 w

ID	Car
1	Ford Club Wagon (1
2	Acura TL (2005)
3	Volkswagen Jetta (2
4	Chevrolet Uplander (2
5	Plymouth Roadrunner
6	GMC Safari (2000
7	Lamborghini Murciélag
8	GMC 3500 (1999

Click Check my progress to verify the objective.

Challenge: Sales summary

Check my progress

Optional challenge

As **optional** challenges, you could try some of the following functionalities:

1. Sort the list of cars in the PDF by total sales.
2. Create a pie chart for the total sales of each car made.
3. Create a bar chart showing total sales for the top 10 best selling vehicles using the ReportLab Diagra library. Put the vehicle name on the X-axis and **total revenue** (remember, price * total sales!) along the Y-axis.

Congratulations!

Congrats! You've successfully written a Python script to automatically generate a PDF and send it through email.

End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.