FDZ Gesundheit

Bundesinstitut für Arzneimittel und Medizinprodukte

Das Antragsportal des Forschungsdatenzentrums Gesundheit

Nutzerhandbuch für die Arbeit mit Posit und JupyterLab

Version 01-04-000

Inhalt

1 Voraussetzungen

Bevor Sie mit der Arbeit in Ihrem Analyseraum beginnen, lesen Sie bitte die Programmcoderichtlinien, die sich in der Anlage zu den Nebenbestimmungen befinden. **Wichtig! Wenn Sie diese Regeln nicht einhalten, kann nicht sichergestellt werden, dass die Ausgabe Ihrer Ergebnisse mit Ihren erstellten Python-Skripten ermöglicht wird.**

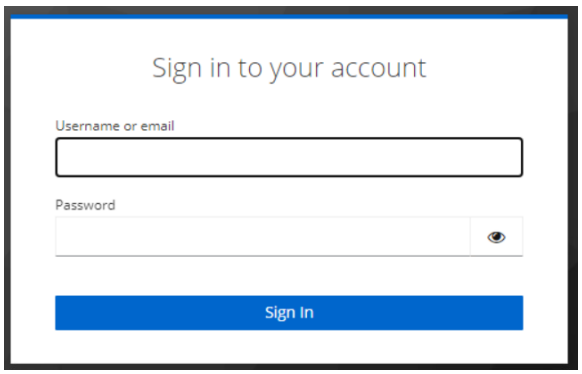
2 Posit Workbench

Die Posit Workbench ist die zentrale Plattform, auf der Sie Ihre Analyse Sessions verwalten können. Sie können - je nach dem, welche Sprache Sie im Antrag ausgewählt haben - RStudio oder JupyterLab Sessions starten, pausieren oder schließen. Wenn Sie in Ihrem Antrag mit einem Kollegen bzw. Kollegin zusammenarbeiten, können Sie über die Workbench Sessions gemeinsam verwalten oder in individuellen Sessions arbeiten. Zugang zur Workbench erhalten Sie über das Antragsportal.

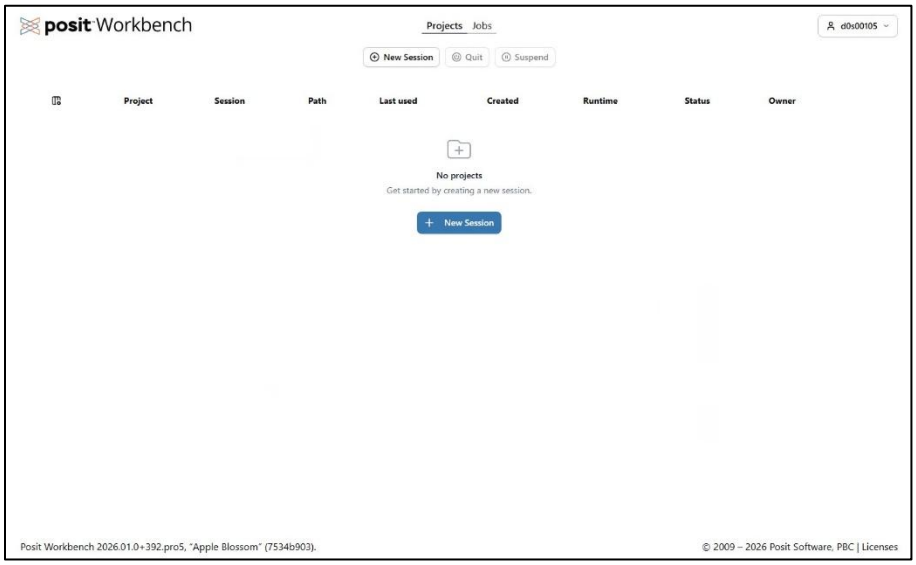
2.1 Einloggen in die Workbench

Klicken Sie auf "Sign in with OpenID.

Tragen Sie Ihren Anmeldenamen sowie Passwort ein (Nutzername und Passwort, das Sie sich selbst nach Freischaltung des Analyseraums vergeben haben). **Diese sollten Ihnen per E-Mail zugesendet worden sein.**

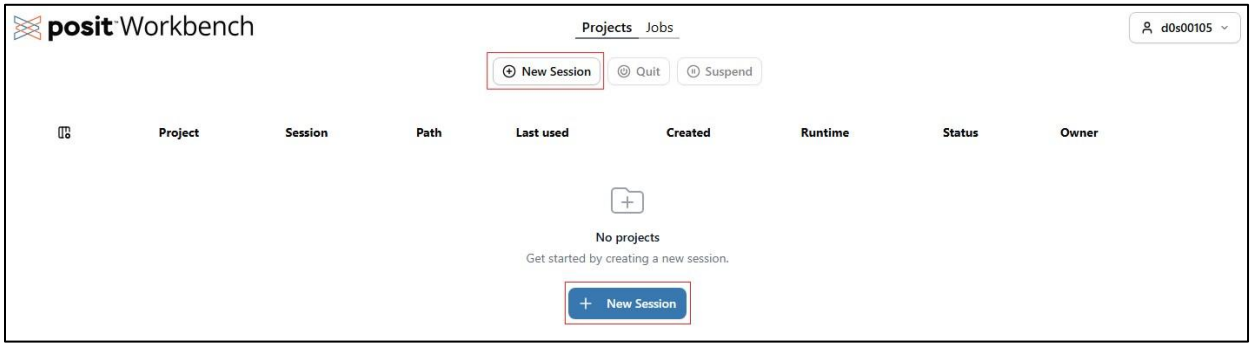


Startseite der Workbench öffnet sich.

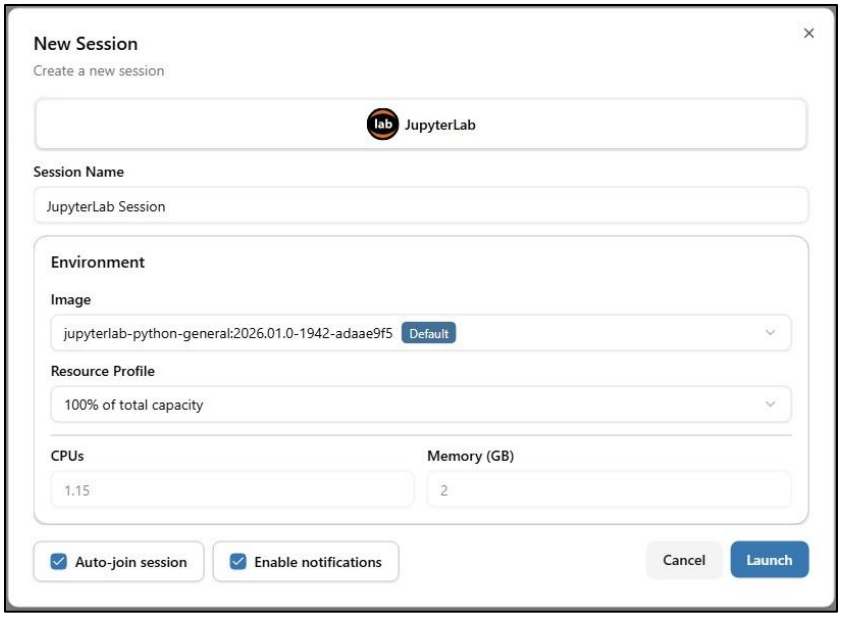


2.2 Starten neuer Sessions

Sie können in Ihrer Workbench JupyterLab Sessions starten, pausieren und schließen. Wenn Sie auf „+ **New Session**“ klicken, können Sie eine neue Session starten.

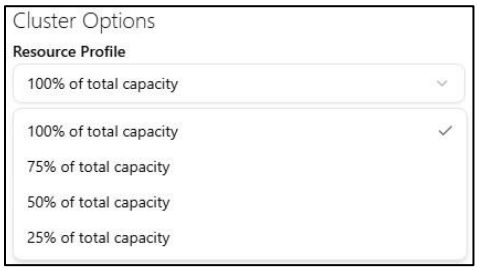


Anschließend erscheint ein Pop up Fenster mit einigen Einstellungen für Ihre Session:

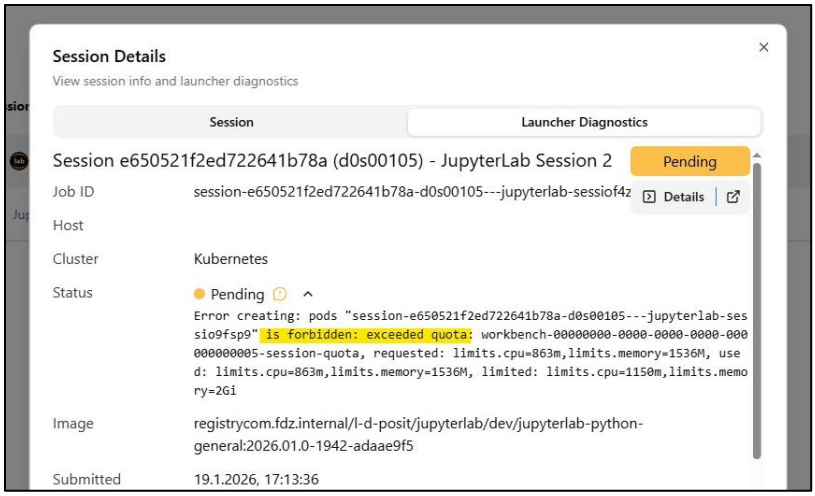


Im Feld "**Session Name**" können Sie jeder Session einen individuellen Namen geben, um sie leichter zu identifizieren, beispielsweise bei der parallelen Ausführung mehrerer Skripte.

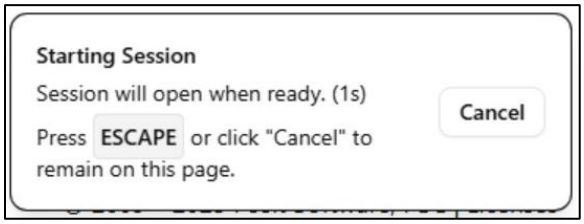
Unter "**Resource Profile**" legen Sie den maximalen CPU-Leistungsanteil für diese Session fest. Es stehen vier Ressourcen-Profile (25 %, 50 %, 75 % und 100 % der Gesamtkapazität) zur Verfügung. Die Auswahl von 100% (Standardeinstellung) ist sinnvoll, wenn Sie ausschließlich in dieser Session arbeiten. Bei mehreren parallelen Analysen (etwa durch weitere Nutzer im Antrag) empfiehlt sich eine Verteilung der Ressourcen, um die Performance zu optimieren.



Hinweis: Bitte beachten Sie, dass sich die Auswahl der Ressourcen auf die gesamte Workbench bezieht. Wenn Sie also zu zweit in diesem Analyseraum tätig sind und ein Kollege bzw. eine Kollegin von Ihnen bereits eine aktive Session mit 50% hat, können Sie in Ihrer Session maximal 50% auswählen. Sollten Sie dennoch versuchen, eine Session mit höherem Ressourcenanteil zu starten, obwohl die verfügbaren Ressourcen bereits vollständig ausgelastet sind, kann die Session nicht gestartet werden und verbleibt im Status "Pending". Über die Option "Details" können die Session Informationen eingesehen werden. Im Tab "Launcher Diagnostics" lassen sich durch das Aufklappen des Status "Pending" (Pfeilsymbol) die Detailmeldungen anzeigen, die den Grund für den ausstehenden Start erläutern (z. B. exceeded quota).



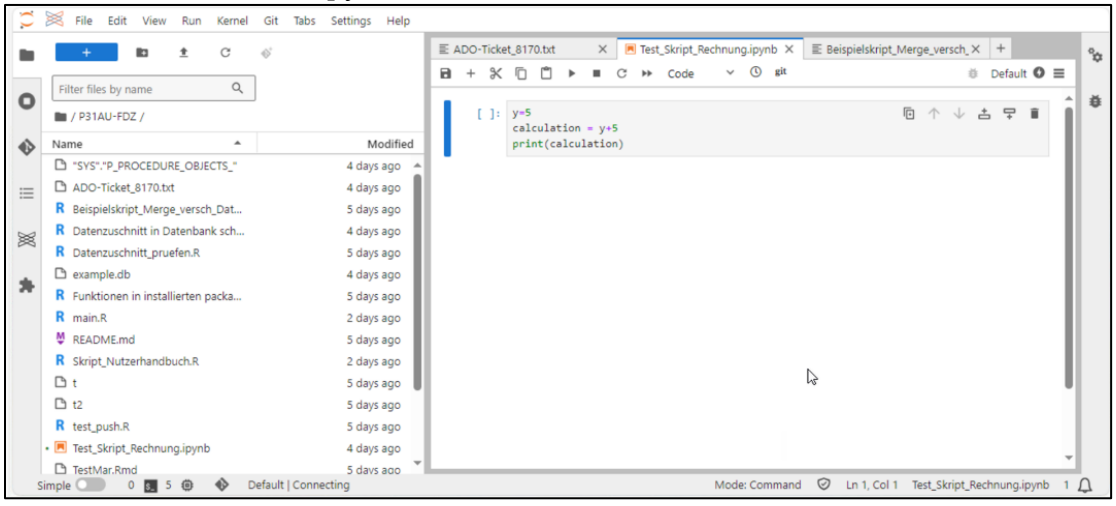
Klicken Sie auf "Launch".
Anschließend wird Ihre Session automatisch gestartet. Das kann einen Augenblick dauern.



3 Erste Schritte mit JupyterLab

3.1 JupyterLab Fenster

Für Einsteiger in JupyterLab empfehlen wir, neben unseren kurzen Erläuterungen, das von JupyterLab bereitgestellte Handbuch zu lesen.
Im Folgenden stellen wir Ihnen die verschiedenen Registerkarten vor, damit Sie alle Funktionalitäten von JupyterLab nutzen können.



3.1.1 Obere Menüleiste

Die obere Menüleiste in JupyterLab enthält Menüs der obersten Ebene, die die in JupyterLab verfügbaren Aktionen mit ihren Tastaturkürzeln anzeigen. Die Standardmenüs sind:

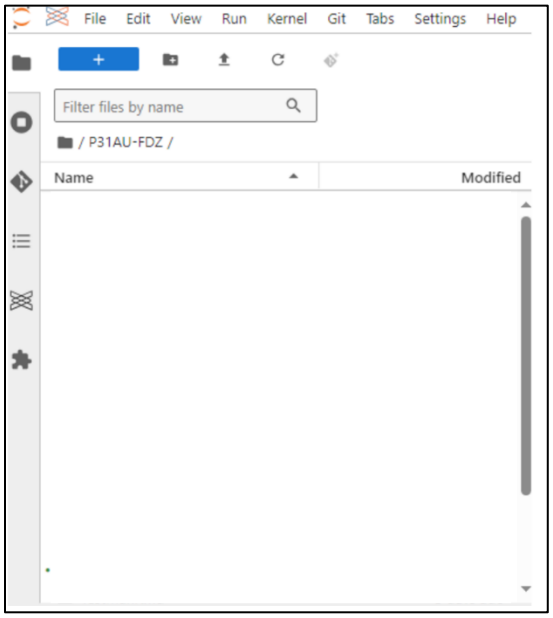
- **File**: Aktionen im Zusammenhang mit Dateien und Ordnern
- **Edit**: Aktionen im Zusammenhang mit der Bearbeitung von Dokumenten und anderen Aktivitäten
- **View**: Aktionen, die das Erscheinungsbild von JupyterLab ändern
- **Run**: Aktionen zum Ausführen von Code in verschiedenen Aktivitäten wie Notebooks und Code-Konsolen
- **Kernel**: Aktionen zum Verwalten von Kerneln, bei denen es sich um separate Prozesse zum Ausführen von Code handelt
- **Git:** bietet schnellen Zugriff auf häufig verwendete Git-Befehle. Für mehr Funktionen zur Versionskontrolle mit Git siehe Kapitel 4 Arbeiten mit GitLab
- **Tabs**: eine Liste der geöffneten Dokumente und Aktivitäten im Dock-Panel
- **Settings**: allgemeine Einstellungen und ein erweiterter Einstellungseditor
- **Help**: eine Liste mit Hilfelinks zu JupyterLab und dem Kernel

3.1.2 Linke Menüleiste

Die linke Menüleiste enthält eine Reihe häufig verwendeter Registerkarten, darunter:

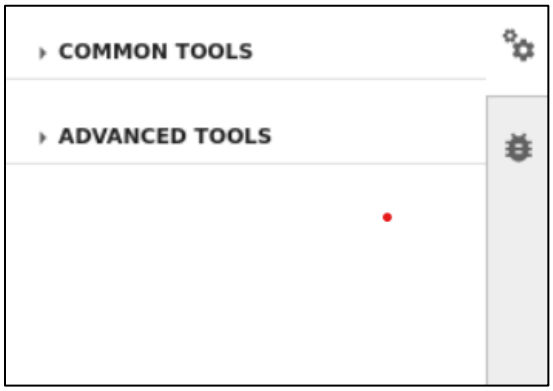
-  Der Dateibrowser zeigt alle Dateien an, die sich in Ihrem Homeverzeichnis (Arbeitsverzeichnis) befinden
-  Diese Registerkarte bietet eine Liste der offenen Tabs im Hauptarbeitsbereich und aller aktuell laufenden Kernels und Terminals. Ein **Kernel** ist der Rechenkern, der hinter einem Jupyter-Notebook läuft. Er ist verantwortlich für das Ausführen von Code in einer bestimmten Programmiersprache. Sie müssen für Ihre Arbeit im Antrag diesbezüglich jedoch nichts aktiv beachten. Ein Terminal ist eine Kommandozeile (Shell), die innerhalb von JupyterLab geöffnet werden kann. Sie können über das Terminal z.B. Systembefehle (z.B. Git-Befehle) oder Skripte manuell ausführen, z. B. durch Eingabe von "python script.py".
-  Die Git-Registerkarte bietet eine umfassende Übersicht über Ihr Git-Repository. Sie können hier Änderungen an Dateien sehen, Branches verwalten, Commits durchführen und den Status Ihres Repositories überwachen.
-  Die "Table of content"-Registerkarte wird automatisch in der linken Seitenleiste generiert, wenn Sie ein Notizbuch, eine Markdown-, Latex- oder Python-Datei geöffnet haben. Die Einträge sind anklickbar und scrollen das Dokument bis zur jeweiligen Überschrift.
-  Die "Proxied Server"-Registerkarte hat im Zusammenhang mit dem FDZ keine Funktionalität, daher können Sie diese Registerkarte ignorieren.

- Der Extension Manager hat im Zusammenhang mit dem FDZ keine Funktionalität, daher können Sie diese Registerkarte ignorieren.

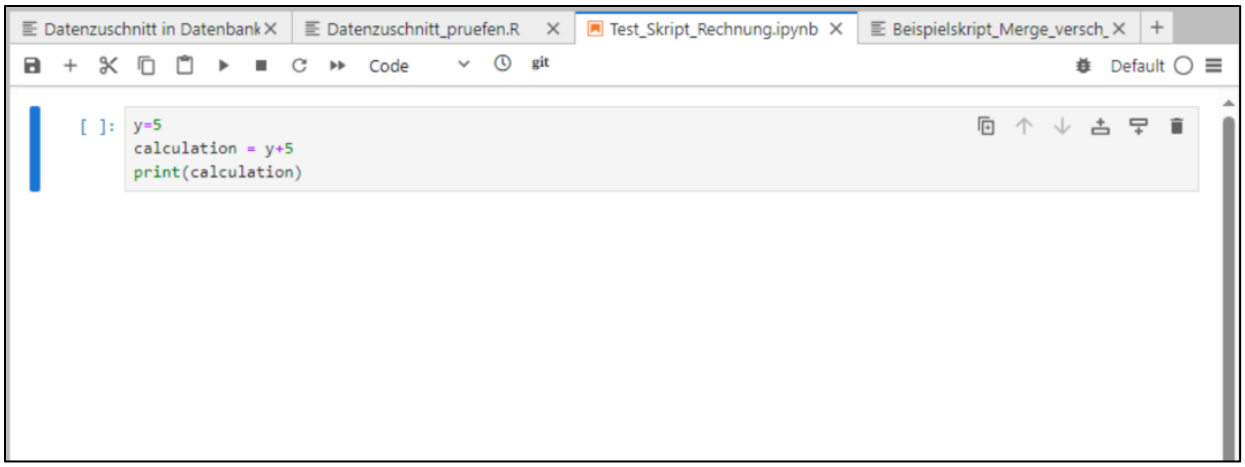


3.1.3 Rechte Menüleiste

- Der Eigenschafteninspektor (aktiv in Notizbüchern) zeigt die Eigenschaften des aktuell ausgewählten Widgets oder Elements an und ermöglicht es Ihnen, diese Eigenschaften direkt zu bearbeiten.
- Der Debugger ermöglicht, Code direkt in Ihre Notebooks, Code-Konsolen und Dateien zu debuggen, z.B. Setzen von Breakpoints, um die Ausführung an bestimmten Stellen zu stoppen und den Zustand des Programms zu untersuchen.



3.1.4 Hauptarbeitsbereich



Der Hauptarbeitsbereich in JupyterLab ist der zentrale Bereich für Ihre Arbeit mit Dokumenten und Aktivitäten. Er bietet unter anderem folgende Funktionen:

- **Dokumente und Aktivitäten**: Der Hauptarbeitsbereich enthält Tabs für verschiedene Dokumente wie Jupyter-Notebooks, Textdateien, Terminals und andere Aktivitäten.
- **Flexible Anordnung:** Sie können Tabs in Panels anordnen und diese in verschiedene Bereiche unterteilen. So können Sie mehrere Dokumente gleichzeitig öffnen und nebeneinander bearbeiten.
- **Drag-and-Drop:** Sie können Tabs einfach per Drag-and-Drop verschieben, um sie neu anzuordnen oder in neue Panels zu ziehen.
- **Resizing**: Passen Sie die Größe der Panels an, um den verfügbaren Platz optimal zu nutzen.
- **Speichern von Workspaces**: JupyterLab ermöglicht es Ihnen, Workspaces zu speichern, die den Zustand Ihrer geöffneten Dateien und Layouts enthalten.

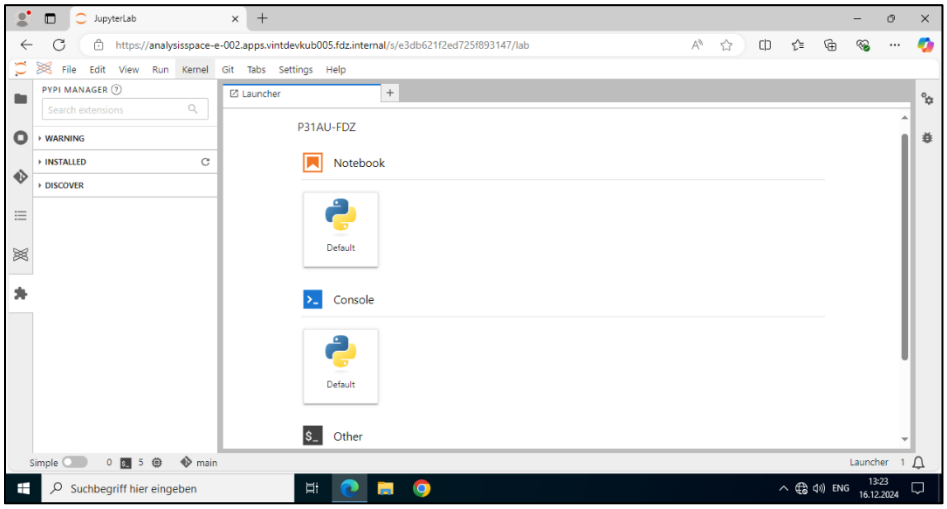
Diese Funktionen machen den Hauptarbeitsbereich zu einem flexiblen und leistungsfähigen Werkzeug für die interaktive und explorative Datenanalyse.

3.2 Erstellen/Verwalten von Skripten

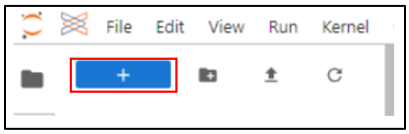
Sie können über JupyterLab Notebooks oder Python-Skripte erstellen.

Bitte beachten Sie, dass kein Einfügen von Inhalten (Strg + C) möglich ist. Wenn Sie Inhalte von beispielsweise alten Skripten nutzen möchten, müssen Sie dieses Skript über das Antragsportal hochladen. Durch den Git Pull Befehl wird dieses Skript in Ihr Homeverzeichnis geladen und steht Ihnen zum Arbeiten zur Verfügung.

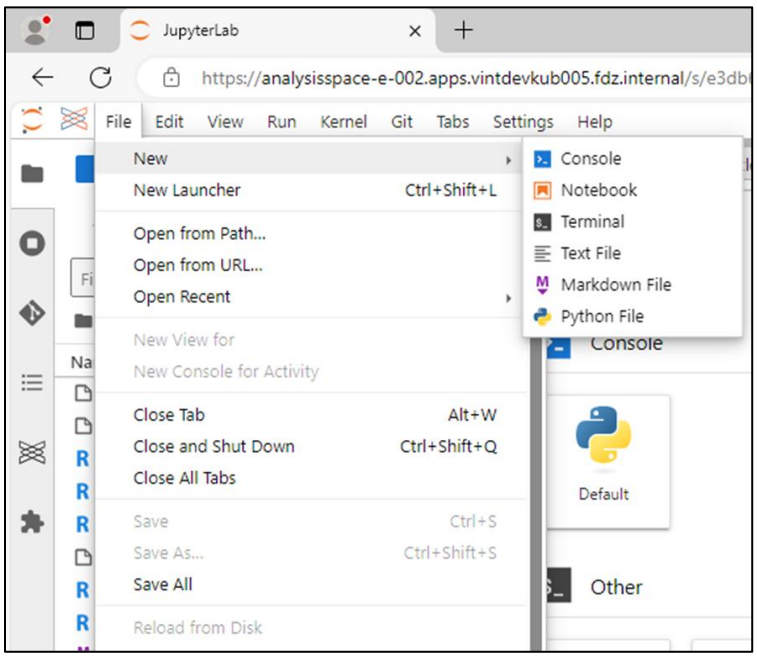
Sie können außerdem Befehle über die Console oder das Terminal eingeben. Zum Öffnen eines neuen Skriptes oder Datei können Sie dies über den Launcher tun, der sich beim erstmaligen Öffnen einer neuen Session in Ihrem Analyseraum öffnet oder über das + Symbol zum Öffnen eines neuen Tabs:



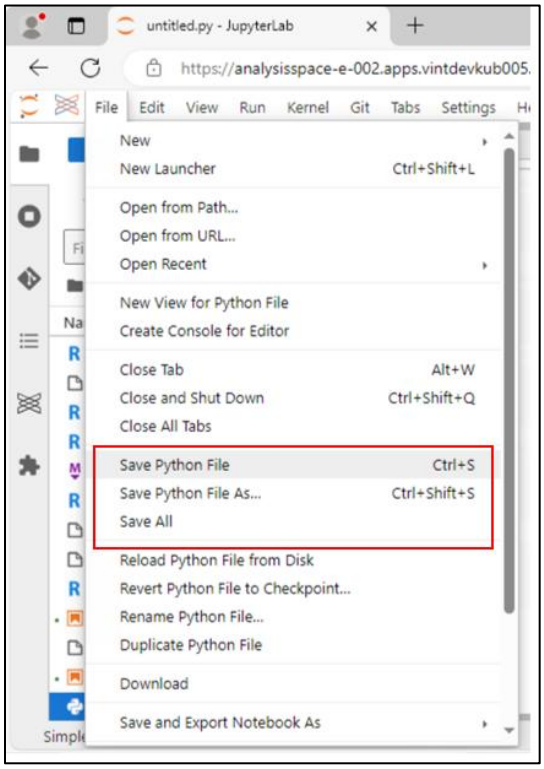
Der Launcher öffnet sich auch, wenn Sie oben links auf das blaue + Symbol klicken:



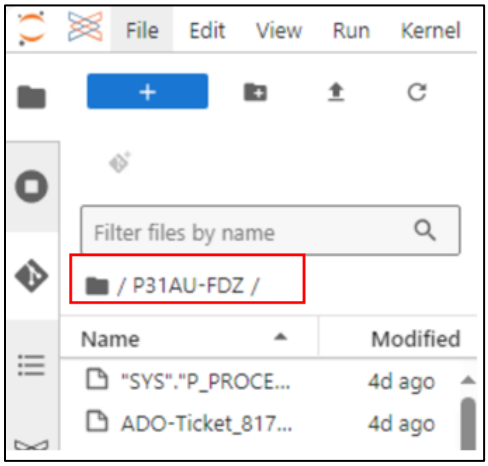
Alternativ können Sie auch über File > New eine Auswahl zum Erstellen einer neuen Datei auswählen:



Zum **Speichern** klicken Sie auf "File" und dann auf "Save Python file" (oder das entsprechende Dateiformat). Sie können in dieser Menüauswahl auch alle geöffneten Dateien speichern (Save All).



Zum **Öffnen** einer bereits gespeicherten Datei, klicken Sie auf die Datei Registerkarte in der linken Menüleiste und stellen sicher, dass Sie sich in Ihrem Analyseraum-Ordner befinden. Das stellen Sie fest, indem hinter dem Dateiname der Name Ihres Analyseraums steht.



Sollte der Name hier nicht stehen, klicken Sie bitte auf den Ordner in der darunter befindlichen Auflistung, der den Namen Ihres Analyseraums trägt.

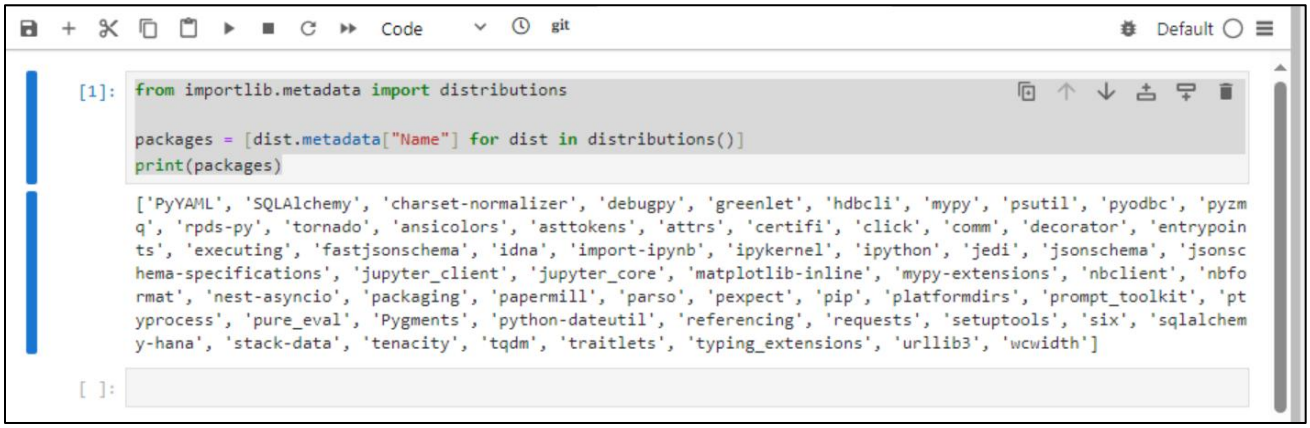
3.3 Arbeiten mit Packages/Paketen im FDZ

Aufgrund der sehr hohen Sicherheitsanforderungen an das System steht derzeit nur ein begrenztes Angebot an Packages (Paketen) zur Verfügung. Diese Packages sind bereits vorinstalliert und stehen Ihnen somit in JupyterLab zur Verfügung.

Wenn Sie eine Liste dieser vorinstallierten Packages erhalten möchten, können Sie folgenden Befehl verwenden:

Liste Pakete Python

```
from importlib.metadata import distributions

packages = [dist.metadata["Name"] for dist in distributions()]
print(packages)
```



Sollten Sie nach einer bestimmten Funktionalität suchen, die ein Paket beinhaltet, welches nicht vorinstalliert ist, prüfen Sie einmal die vorgeschlagene Schritte in den FAQ.

4 Arbeiten mit GitLab

4.1 Einführung in GitLab

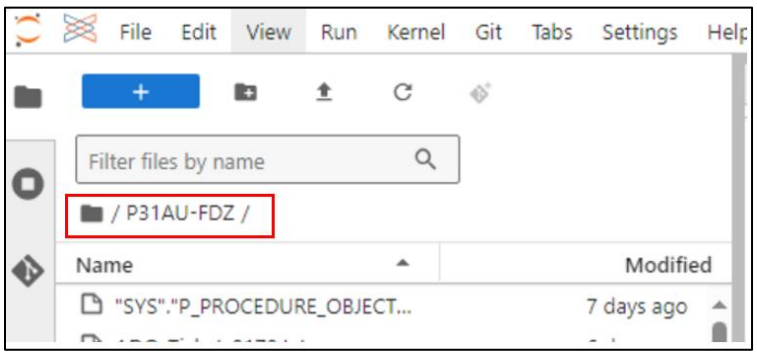
Im FDZ wird GitLab zum Verwalten Ihrer Analyseskripte verwendet. Mithilfe Ihres zugewiesenen Git Repositorys können Sie Ihre Skripte hochladen und sichern. Mit der Push-Funktion können Sie Ihre Skripte nicht nur sichern, sondern auch mit anderen Nutzern und Nutzerinnen in Ihrem Analyseraum teilen. Auf diese Skripte können auch FDZ-Mitarbeitende zugreifen, um Sie zum Beispiel in Supportanfragen zu unterstützen. Außerdem muss das Skript, das Sie für die Lauffähigkeitsprüfung sowie für die Zwischen- und Endergebnisse verwenden möchten, im Git Repository gespeichert werden.

Das Abspeichern der Skripte im Git Repository ist sehr wichtig, um sicherzustellen, dass Ihre Skripte zu jedem Zeitpunkt gesichert sind. **Daher sollten Sie den aktuellen Stand Ihrer Skripte nach Möglichkeit immer ins Git Repository pushen.** Wie das funktioniert, erfahren Sie in den nachfolgenden Kapiteln.

4.2 Automatischer GitLab Zugriff in JupyterLab

Wenn Sie eine JupyterLab Session öffnen, sollte die Git Anbindung automatisch hergestellt sein. Über die Git Registerkarte in der linken Menüleiste können Sie mit Ihrem Git Repository interagieren. Die Verbindung mit dem Git Repository ist jedoch nur möglich, wenn Sie sich in Ihrem Projektordner befinden, der nach Ihrem Analyseraum benannt ist.

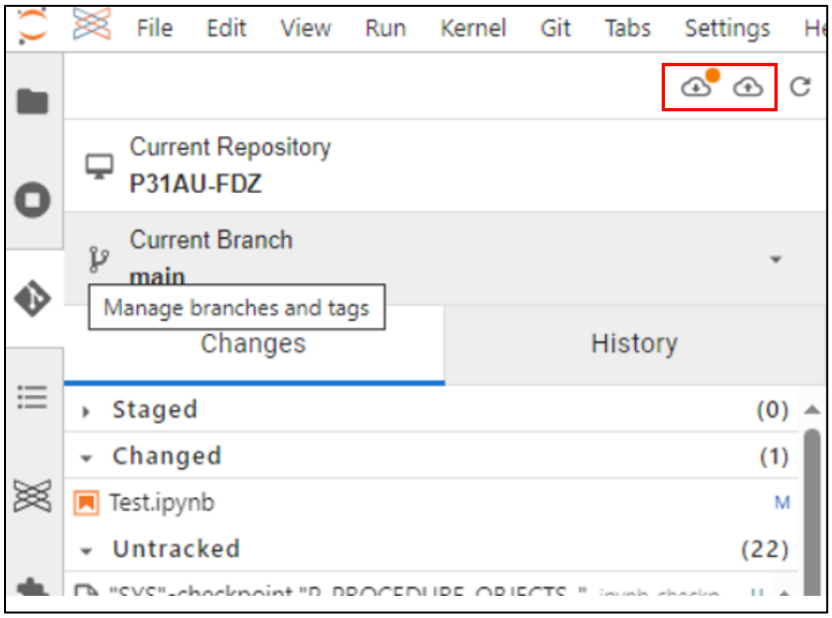
Sie sollten somit zu jedem Zeitpunkt sicherstellen, dass Sie in der Datei Registerkarte den Projektordner geöffnet haben:



4.3 Arbeiten mit dem Git Repository

Sollten Sie noch nie mit GitLab in JupyterLab gearbeitet haben, können Sie eine Anleitung auf der entsprechenden GitLab Seite finden: [Use JupyterLab's GitLab Extension]

Wir zeigen Ihnen hier die gängigen Anwendungen, denen Sie im Rahmen Ihrer Arbeit im FDZ begegnen werden.



Oben im Screenshot sehen Sie zwei kleine Wolken. Wenn Sie auf die **linke Wolke** klicken, pullen Sie den aktuellen Stand aus Ihrem Git Repository. Mit der **rechten Wolke** führen Sie den Push Befehl aus. Nähere Informationen zu den Push Möglichkeiten erfahren Sie in der folgenden Beschreibung dieser Git Registerkarte:

- Unter **Current Repository** können Sie sehen, dass Sie sich in Ihrem GitLab Repository befinden.
- Das Feld **Current Branch** zeigt Ihnen an, in welchem Branch Sie sich befinden. Wenn Sie zu zweit an Ihrem Antrag arbeiten, sollten Sie sich absprechen, damit nicht zwei Nutzende zeitgleich an derselben Datei arbeiten, da das zu **Merge-Konflikten** führen kann. Daher sollten Sie regelmäßig den aktuellen Stand des Git Repositorys pullen. Wenn Sie aktiv an einem Skript arbeiten, kann es hilfreich sein, kleine Änderungen häufiger zu committen und zu pushen, anstatt große Änderungen auf einmal zu machen. Das erleichtert das Zusammenführen von Änderungen und das Erkennen von Konflikten.
- Die Registerkarte **Changes** zeigt Ihnen alle Änderungen an, die Sie an den Dateien vorgenommen haben, bevor Sie diese speichern (committen). Sie können sehen, welche Dateien geändert, hinzugefügt oder gelöscht wurden und welche konkreten Änderungen an den Dateien vorgenommen wurden.

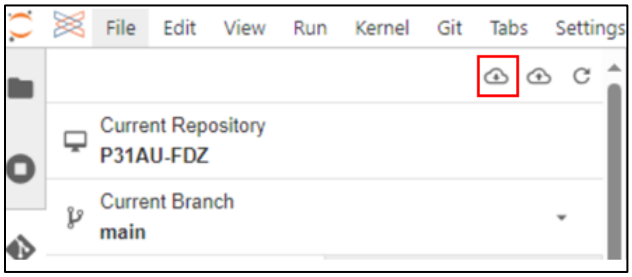
    - **Staged (gestaged)**: Dateien, deren Änderungen Sie für den nächsten Commit vorbereitet haben. Diese Änderungen werden in einem sogenannten **Staging-Bereich** gespeichert. Sobald Sie diese Dateien **committen**, werden die gestagten Änderungen in das Git-Repository übernommen.
    - **Changed (nicht gestaged)**: Dateien, an denen Sie Änderungen vorgenommen haben, aber diese Änderungen wurden noch nicht für den Commit vorgemerkt. Git weiß, dass sich etwas geändert hat, aber Sie müssen die Datei explizit zum Staging-Bereich hinzufügen, um diese Änderungen im nächsten Commit zu speichern.
    - **Untracked**: Dateien, die noch nicht vom Git-Tracking erfasst wurden. Sie existieren im Arbeitsverzeichnis, aber Git ignoriert sie noch.
- Im Feld **Summary** können Sie eine kurze Beschreibung Ihrer Commit Nachricht verfassen
- Das Feld **Description** ist optional. Dort können Sie ergänzende Informationen eintragen, die Ihnen oder Ihren Kollegen bzw. Kolleginnen helfen, die Versionierung Ihrer Skripte nachvollziehen zu können.

Im Folgenden können Sie sehen, wie die Schritte von der Erstellung eines Skriptes bis zum erfolgreichen Pushen eines Skriptes in Ihr Git Repository ablaufen könnte:

4.3.1 Git Pull

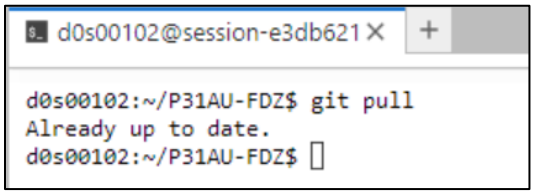
Der Befehl „Pull“ dient in Git zum Lesen aus dem Git Projekt Repository. Dieser Befehl ist insbesondere wichtig, wenn Sie zusammen mit jemandem in einem Git Repository arbeiten. Das stellt sicher, dass Sie immer die aktuellen Dateien in Ihr Homeverzeichnis zum Bearbeiten in Ihrer Session zur Verfügung haben.

Der Button zum Pullen ist eine kleine Wolke, die sich oben innerhalb der Git Registerkarte befindet.



Wenn Sie die Wolke anklicken, sollte unten rechts im Bildschirm eine Meldung auftreten, dass das Pullen erfolgreich war.

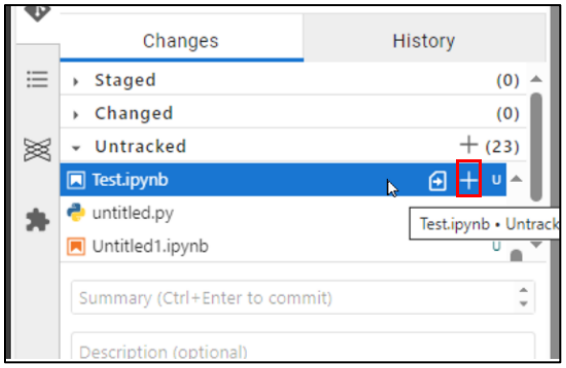
Alternativ können Sie auch im Terminal den Befehl "git pull" eingeben.



4.3.2 Git Commit

Wenn Sie Dateien (z.B. JupyterLab Notebook) bearbeitet haben, dann speichern Sie dieses zuerst ab, indem Sie auf das Speichersymbol oben links im Skript Fenster klicken oder unter File > Save (z.B. Notebook).

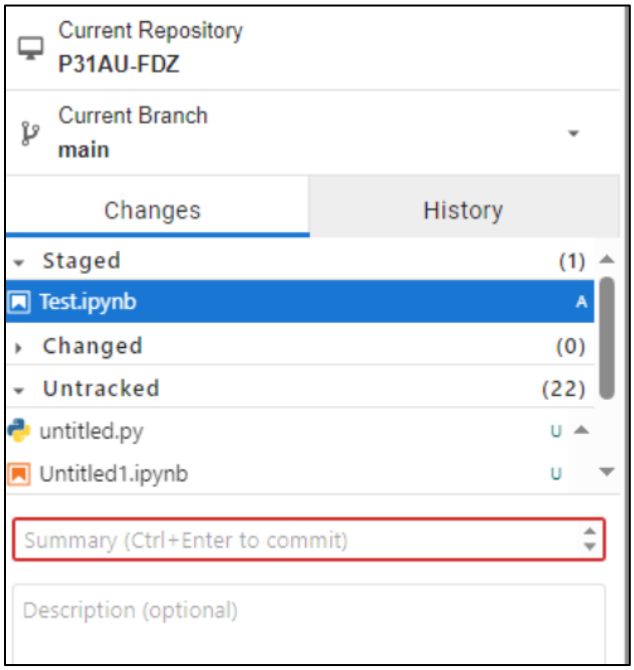
Anschließend öffnen Sie die Git Registerkarte . Wenn Sie ein Skript abgespeichert haben, das **noch nie** in Ihr Git Repository gepushed haben, werden Sie Ihr Skript unter "Untracked" finden.



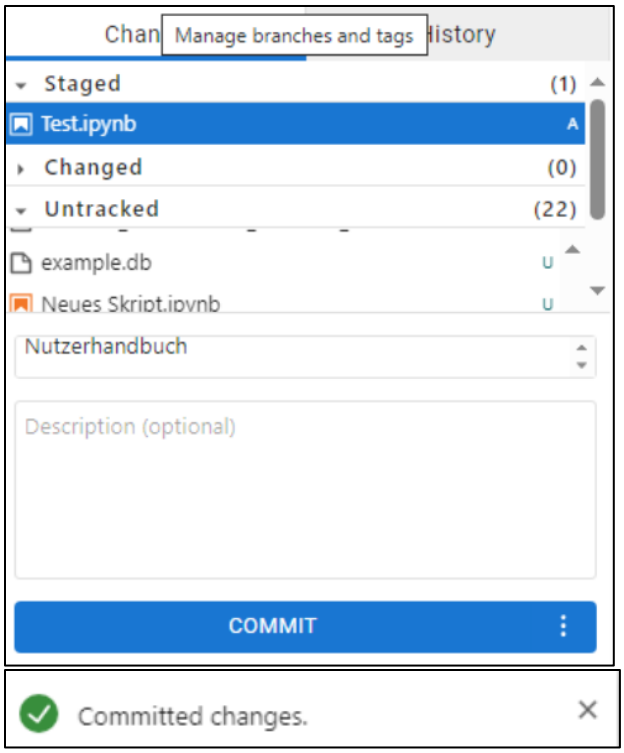
Wenn Sie mit der Maus auf den Namen dieses Skripts steuern, erscheint ein + Symbol. Durch Anklicken dieses Zeichens wird Ihr Skript unter "Staged" aufgeführt.

Wenn Sie ein Skript geändert haben, das in einer alten Version bereits im Git Repository gesichert war, erscheint es unter "Changed" anstelle von "Untracked".

Als nächstes tragen Sie im rot umkreisten Feld "Summary" einen Kommentar zu Ihrem Commit ein. Dieser Kommentar dient Ihnen sowie möglichen anderen Nutzer und Nutzerinnen in Ihrem Analyseraum, die gepushten Änderungen nachvollziehen zu können.



Nach Eintragen des Summary Feldes können Sie weiter unten das Feld "Commit" anklicken. Gegebenenfalls müssen Sie dafür etwas nach unten scrollen.



4.3.3 Git Push

Nachdem Sie die Meldung über das erfolgreiche Commiten der Änderungen erhalten haben, sollte ein orangener Punkt an der rechten Wolke oben rechts in der Registerkarte erscheinen.

Wenn Sie auf diese Wolke klicken, sollten anschließen unten rechts in Ihrem Hauptarbeitsbereich die Meldung über das erfolgreiche Pushen erscheinen

5 Datenbankverbindung

Wir haben bereits eine Datenbankverbindung ("conn") für Sie angelegt. Daher verwenden Sie nur diese Verbindung zur Interaktion mit der Datenbank in Ihrem Analyseraum und für die Lauffähigkeitsprüfung.

5.1 Anzeigen der Datentabellen im Analyseraum

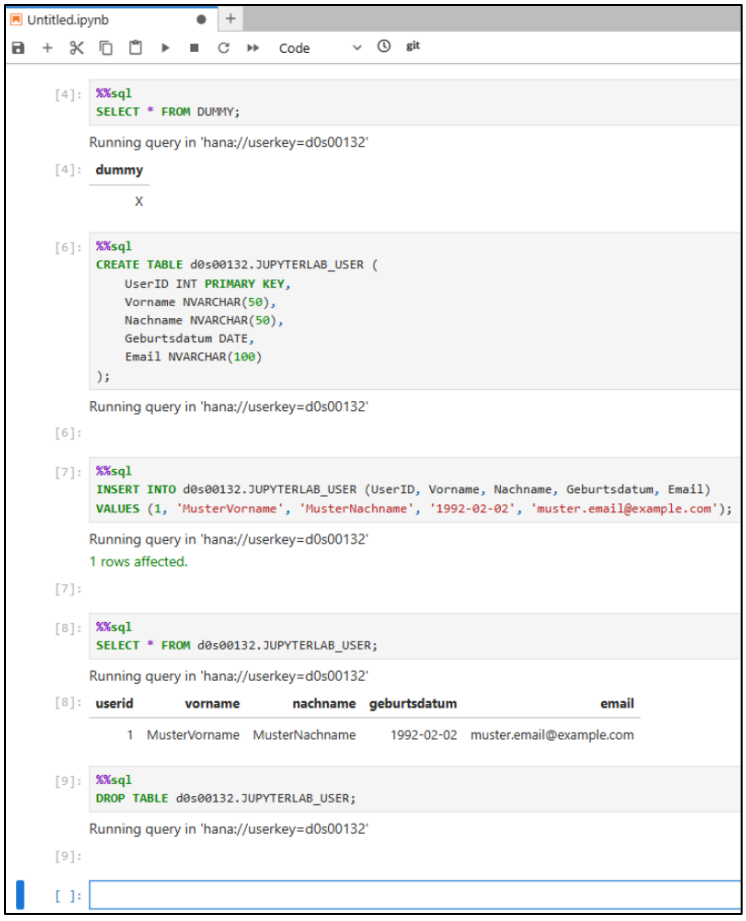
In Ihrem Analyseraum finden Sie verschiedene Datentabellen. Sie finden initial dort Datentabellen mit Ihren Analysedaten, Kataloge des FDZ sowie die Tabellen "fdz.db.table.internal: :ERGEBNIS_LAUFFAEHIGKEIT" und "fdz.db.table.internal: :ERGEBNIS_LAUFFAEHIGKEIT_DETAILS". Die beiden letzten Tabellen werden nach jeder Lauffähigkeitsprüfung mit Informationen zu dieser Prüfung befüllt.

Mit folgendem Code können Sie sich alle Namen der enthaltenen Tabellen in Ihrer Datenbank anzeigen lassen:
(hier am Beispiel der Antragsnummer P31851_61. Diese Nummer müssen Sie an Ihre Antragsnummer anpassen!)

```python
from sqlalchemy import text

# Gemeinsame Abfrage für Tabellen und Views
combined_query = """
SELECT 'TABLE' AS OBJECT_TYPE, TABLE_NAME AS NAME
FROM TABLES
WHERE SCHEMA_NAME = 'P31851_61'

UNION ALL

SELECT 'VIEW' AS OBJECT_TYPE, VIEW_NAME AS NAME
FROM VIEWS
WHERE SCHEMA_NAME = 'P31851_61'
"""

result = conn.execute(text(combined_query))
objects = result.fetchall()

# Ausgabe
for obj_type, name in objects:
    print(f"{obj_type}: {name}")
```

```
TABLE: fdz.db.table.internal::ERGEBNIS_LAUFFAEHIGKEIT
TABLE: fdz.db.table.internal::ERGEBNIS_LAUFFAEHIGKEIT_DETAILS
VIEW: DATRAV_VERS
VIEW: DATRAV_ZAHNBEF
VIEW: DATRAV_ZAHNFALL
VIEW: DATRAV_ZAHNLEIST
VIEW: EBM
VIEW: ERGEBNIS_LAUFFAEHIGKEIT
VIEW: ERGEBNIS_LAUFFAEHIGKEIT_DETAILS
VIEW: ICD_JAHRESVERTEILUNG
VIEW: INKAR
VIEW: INKAR_GEMEINDEN
VIEW: INKAR_KREISE
VIEW: INKAR_METROPOLREGION
VIEW: OPS
VIEW: OSM_PLZ
```

Zusatzinfo: Sie können auch über %%sql SQL-Befehle ausführen, ohne ein Python Code schreiben zu müssen. Weitere Informationen hierzu finden sie unter diesen Link: https://jupysql.readthedocs.io/en/latest/quick-start.html.
Nachfolgend sehen Sie unterschiedliche Beispiele:



5.2 Anzeigen des Datensatzzuschnittes/Inhalt einer Tabelle

In Ihrem Analyseraum haben Sie einen spezifisch für Ihren Antrag zugeschnitten Datenzuschnitt zur Verfügung, mit dem Sie Ihre Analyseskripte entwickeln können. Diesen Datenzuschnitt finden Sie in verschiedenen Datentabellen in Ihrem Schema. Mit folgendem Code sehen Sie, wie Sie sich den Inhalt einer dieser Datentabellen anzeigen lassen können:

```
import pandas as pd

query = """
SELECT *
FROM P31851_28.DATRAV_VERS
LIMIT 10
"""

df = pd.read_sql(query, conn)
df
```

	bjahr	datenmodell	psid
0	2019	3	F0416CE7542F646A19000216E4A6F8DB
1	2019	3	34426CE7542F646A19000216E4A6F8DB
2	2019	3	C4426CE7542F646A19000216E4A6F8DB
3	2019	3	69416CE7542F646A19000216E4A6F8DB
4	2019	3	05426CE7542F646A19000216E4A6F8DB
5	2019	3	09426CE7542F646A19000216E4A6F8DB
6	2019	3	D1426CE7542F646A19000216E4A6F8DB
7	2019	3	F9426CE7542F646A19000216E4A6F8DB
8	2019	3	E6416CE7542F646A19000216E4A6F8DB
9	2019	3	6A426CE7542F646A19000216E4A6F8DB

5.3 Erzeugen und Speichern neu erzeugter Datensätze in die Datenbank

Ergebnisse, die Sie als Zwischenergebnisse bzw. Endergebnisse ausgegeben bekommen möchten, müssen Sie als "RT_"-Tabellen in die Datenbank schreiben. Dafür müssen Sie erzeugte Tabellen zuerst als Datentabelle abspeichern (1. + 2.) und anschließend in die Datenbank schreiben (4.).

1. Erstellen einer leeren Datentabelle

```
#Erzeugen einer leeren Datentabelle

from sqlalchemy import text

create_table_query = """
CREATE COLUMN TABLE P31851_87.RT_HAEUFIGKEIT (
CNT_HAEUFIGKEIT INTEGER
)
"""
conn.execute(text(create_table_query))
conn.commit()
```

2. Befüllen dieser Datentabelle

```
insert_query = """
INSERT INTO P31851_87.RT_HAEUFIGKEIT (CNT_HAEUFIGKEIT)
SELECT COUNT(*) AS CNT_HAEUFIGKEIT
FROM P31851_87.DATRAV_VERS
"""
conn.execute(text(insert_query))
conn.commit

print("Tabelle RT_HAEUFIGKEIT wurde erfolgreich erstellt und befüllt")
```

3. Anzeigen lassen dieser Tabelle

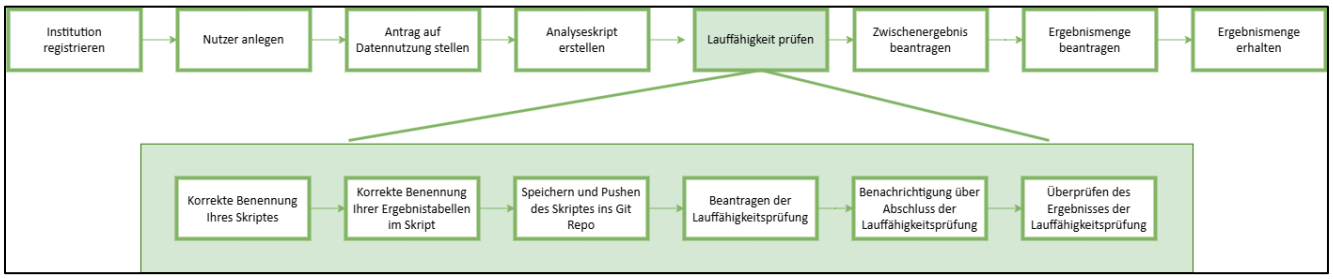
```
import pandas as pd
query_RT = """
SELECT * FROM P31851_87.RT_HAEUFIGKEIT
"""
df_RT = pd.read_sql(query_RT, conn)
df_RT
```

	cnt_haeufigkeit
0	2225150

4. Datentabelle in Datenbank schreiben

```
df_RT.to_sql(
        name="RT_HAEUFIGKEIT",
        con=conn,
        schema="P31851_123",
        if_exists='replace',
        index=False
)
```

6 Prozess Lauffähigkeitsprüfung



Die Prüfung der Lauffähigkeit Ihrer Skripte ist ein sehr wichtiger Arbeitsschritt in Ihrer Antragsbearbeitung. Wie Sie wissen, entwickeln Sie Ihre Skripte - je nach Antrag - entweder auf einem synthetischen Datensatz, der auf Echtdaten basiert, oder auf einem stark pseudonymisierten Datenauszug. Für die Erzeugung Ihrer echten Endergebnisse ist es notwendig, dass Ihre Skripte auch auf dem Echtdatensatz ausgeführt werden können.

Sie können die Lauffähigkeit Ihrer Skripte bis zu 10x überprüfen lassen. Bitte beachten Sie hierbei, dass Sie für die Ausgabe Ihrer Zwischenergebnisse sowie Endergebnisse erst eine Lauffähigkeitsprüfung Ihres Skriptes absolvieren müssen. Das bedeutet, dass die Zwischenergebnisse und auch Endergebnisse auf dem letzten Skript, welches für die Lauffähigkeitsprüfung verwendet wurde, basieren.

Im Folgenden erklären wir Ihnen, welche Schritte Sie befolgen müssen, damit die Lauffähigkeit Ihrer Skripte erfolgreich durchlaufen.

6.1.1 Korrekte Benennung Ihres Skriptes

Wie bereits in den Programmcoderichtlinien erklärt, müssen Sie Ihr Skript, welches Sie für die Erzeugung von Zwischen- und Ergebnismengen verwenden wollen, je nach ausgewählter Programmiersprache **main.ipynb/main.py** benennen. Die korrekte Benennung ist für die Ausführung der Lauffähigkeitsprüfung relevant, da es sich um einen automatisierten Prozess handelt und kein Skript ausgewählt wird, welches nicht diesen Namen besitzt. Daher stellen Sie sicher, dass alle Analysen, die Sie für die Lauffähigkeitsprüfung bzw. Zwischen-/Endergebnisse verwenden möchten, entweder im main Skript enthalten sind oder alternativ entsprechende Befehle vorhanden sind, um Analysen aufzurufen, die in einem anderen Skript aus dem Repository enthalten sind (z.B. einem SQL Skript).

6.1.2 Korrekte Benennung Ihrer Ergebnistabellen

Die Ergebnistabellen, die Sie mit Ihrem Skript erzeugen, müssen ebenfalls einer Namenskonvention entsprechen. Nur erzeugte Tabellen, die das Präfix "RT_" besitzen, können als Zwischen- oder Endergebnisse ausgegeben werden. Daher ist zu empfehlen,

diese Namenskonventionen bereits in Ihren Skripten, die Sie für die Lauffähigkeitsprüfung verwenden, umzusetzen. Bitte beachten Sie, dass diese Tabellen in die Datenbank geschrieben werden müssen und nicht ins Environment.

Erstellen und Abspeichern von Tabellen zur Beschreibung der Ergebnistabellen (DD Tabellen)

Die Beschreibung der Ergebnismenge muss innerhalb von sog. Data Description Tabellen (DD Tabellen) erfolgen. Das von Ihnen eingereichte Skript muss zwei Tabellen mit dem Präfix DD_ erzeugen, welche unten dargestellte Form haben:

6.1.3 DD_RESULT_TABLES (Beispiel)

Tabellen name	Klassifikation_Versichertenbezug_ Leistungstraegerbezug	Klassifikation_ statistisch	Beschreibung	Einschluss_ Filterkriterien
RT_1	Versichertenbezug	Mittelwert	Kennzahl: Mittelwert des Alters der betrachteten Versicherten	Nur Versicherte der Geburtsjahrgänge 1980-1989
RT_2	Versichertenbezug	Häufigkeitstabelle	Kreuztabelle aus Altersgruppe und Geschlecht	Alle beantragten Versicherten

6.1.4 DD_RESULT_COLS (Beispiel)

Tabellenname	Klassifikation_statistisch	Spaltenname	Beschreibung
RT_1	Mittelwert	MW_Alter	Mittelwert des Alters der betrachteten Versicherten
RT_1	Fallzahl	CNT_D_PSID	Fallzahl der Versicherten
RT_2	Häufigkeitstabelle	Altersgruppe	Altersgruppe der betrachteten Versicherten
RT_2	Häufigkeitstabelle	Geschlecht	Geschlecht der betrachteten Versicherten
RT_2	Fallzahl	CNT_D_PSID	Fallzahl der Versicherten

Beispielskripte zum Erstellen dieser Tabellen finden Sie im Homeverzeichnis Ihres Analyseraums im Order "Hilfen".

6.1.5 Speichern und Pushen Ihres Skriptes ins Git Repository

Damit Ihr Skript für die Lauffähigkeitsprüfung verwendet werden kann, müssen Sie dieses in der Version, die Sie verwenden wollen, in Ihr Git Repository pushen. Dafür gehen Sie bitte wie im Kapitel 4. Arbeiten mit GitLab vor.

6.1.6 Beantragen der Lauffähigkeitsprüfung im Antragsportal

Sie haben nun alle relevanten Schritte in Ihrem Analyseraum vorgenommen, um eine Lauffähigkeit Ihres Skriptes beantragen zu können. Die Lauffähigkeitsprüfung können Sie nun im Antragsportal beantragen.

6.1.7 Einsehen der Ergebnismeldung Ihrer Lauffähigkeitsprüfung

Wenn Ihre Lauffähigkeitsprüfung abgeschlossen wurde, erhalten Sie eine E-Mail. Sie können das Ergebnis Ihrer Lauffähigkeitsprüfung in Ihrem Analyseraum einsehen. Diese finden Sie in einer entsprechenden Tabelle mit der Bezeichnung "ERGEBNIS_LAUFFAEHIGKEIT" sowie "ERGEBNIS_LAUFFAEHIGKEIT_DETAILS" (s. Beispielcode):

```
#Anzeigen der Informationen zur Lauffähigkeitsprüfung
import pandas as pd
query = """
SELECT*
FROM P31851_10.ERGEBNIS_LAUFFAEHIGKEIT
"""
df_LFP = pd.read_sql(query, conn)
df_LFP

#Anzeigen der Tabelle mit mehr Details
query_details = """
SELECT*
FROM P31851_10.ERGEBNIS_LAUFFAEHIGKEIT_DETAILS
"""
df_LFP_details = pd.read_sql(query_details, conn)
df_LFP_details
```

[4]:

	ausfuehrungs_id	rt_objekt	spaltennamen	anzahl_zeilen	anzahl_spalten	warnungen
0	1	RT_HAEUFIGKEIT	b'["CNT_HAEUFIGKEIT"]'	0	1	

Sollte ein technischer Fehler bei Ihnen angezeigt werden, dann wird Ihnen keinen Versuch der Lauffähigkeitsprüfung abgezogen und Sie können diese erneut beantragen. **Stellen Sie vor der erneuten Beantragung jedoch in jedem Fall sicher, dass sich Ihr main.py oder main.ipynb Skript im Git Repository befindet.**

7 Anfordern und Einsehen der Zwischenergebnisse

Nach erfolgreicher Ausführung einer Lauffähigkeitsprüfung haben Sie bis zu 5x die Möglichkeit, sich Zwischenergebnisse ausgeben zu lassen. In diesem Fall werden Ihnen die während der Lauffähigkeitsprüfung erzeugten RT_-Tabellen in der Datenbank in Ihrem Analyseraum zur Verfügung gestellt.

Die Datentabellen der Zwischenergebnisse haben ein Präfix, das sich aus ZE und der Version der Zwischenergebnisse zusammensetzt (z.B. ZE01 bei der ersten Ausgabe von Zwischenergebnissen). Um die genaue Bezeichnung herauszufinden, um sich diese ausgeben zu lassen, verwenden Sie z.B. den oben aufgeführten Code zum Anzeigen der enthaltenen Datentabellen im Analyseraum.

In diesem Beispiel hat das Zwischenergebnis die Bezeichnung "ZE01_RT_HAEUFIGKEIT_10e58dd8":

```
TABLE: DD_RESULT_COLS
TABLE: DD_RESULT_TABLES
TABLE: RT_HAEUFIGKEIT
TABLE: ZE01_RT_HAEUFIGKEIT_10e58dd8
```

Mit folgendem Befehl können Sie sich den Inhalt dieser Tabelle anzeigen lassen:

```
import pandas as pd

query = '''
SELECT *
FROM P31851_28."ZE01_RT_HAEUFIGKEIT_10e58dd8"
'''

df = pd.read_sql(query, conn)
df
```

Hinweis: Vielleicht haben Sie festgestellt, dass sich dieser Code zum Anzeigen der Datentabelle etwas unterscheidet zu dem im oberen Kapitel aufgeführten Code zum Anzeigen einer Datentabelle. Das liegt daran, dass bei den Zwischenergebnissen eine ID an die Tabelle angehangen wird, die kleine Buchstaben beinhaltet. In SQL müssen solche Namen gesondert behandelt werden. Daher empfehlen wir, selbst erzeugte Tabellen immer groß zu schreiben bzw. in Großbuchstaben abzuspeichern, um mögliche Probleme zu verhindern.

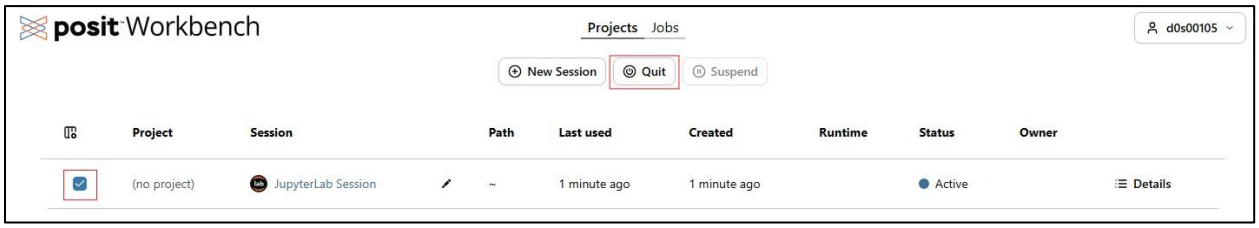
8 Beenden und Management von Sessions

Wenn Sie Ihre Arbeit in Ihrer aktuellen Session beenden wollen, gelangen Sie über  oben links im Fenster zurück auf Ihre Workbench.

In der Übersicht Projects / Jobs sehen Sie alle vorhandenen Sessions. Um eine Session später fortzusetzen, können Sie weiterhin einfach auf den Namen der Session klicken.

Sollten Sie eine aktive Session schließen wollen, weil diese beispielsweise nur für eine bestimmte Analyse laufen sollte, gehen Sie nun wie folgt vor:

1. Markieren Sie die gewünschte Session, indem Sie das Kontrollkästchen links neben der Session auswählen.
2. Klicken Sie anschließend oben auf "**Quit**"

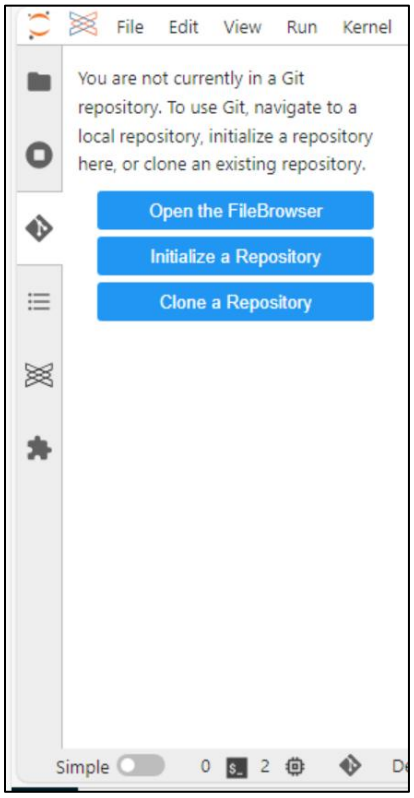


Die ausgewählte Session wird dadurch beendet und die zugehörigen Ressourcen werden freigegeben.

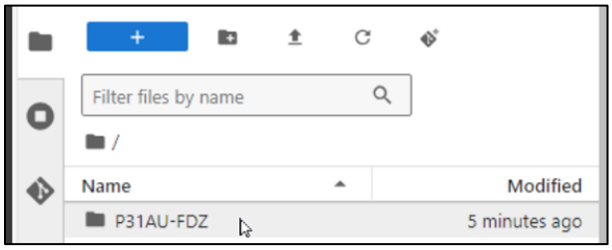
9 FAQ

9.1 Ich kann nicht auf mein Git Repository zugreifen

Sollte Ihnen angezeigt werden, dass Sie sich in keinem Git Repository befinden (s. Bild)



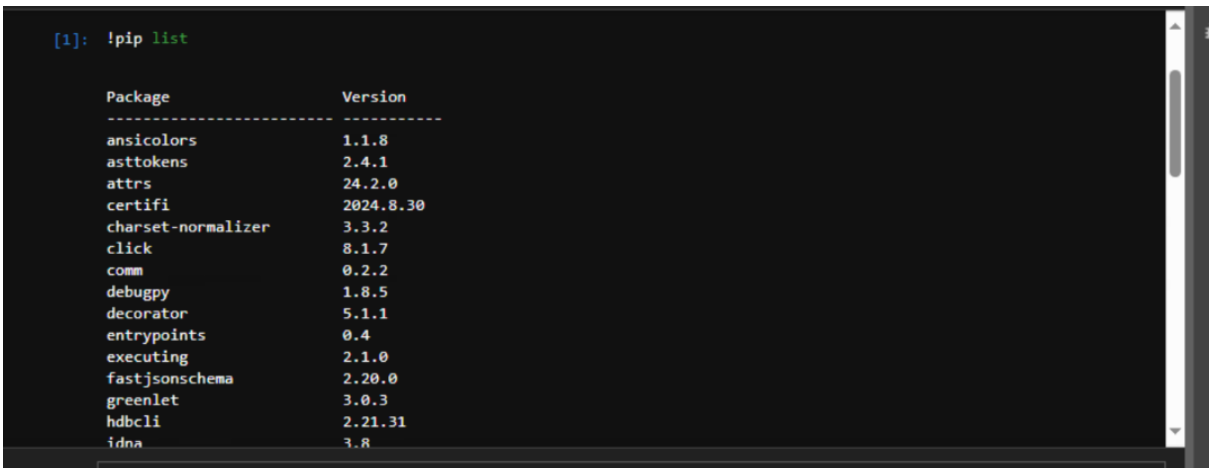
klicken Sie auf die Registerkarte Dateibrowser  und wählen dann den Ordner aus, der nach Ihrem Analyseraum benannt ist.



Wenn Sie anschließend auf die Git Registerkarte klicken, sollte die Verbindung hergestellt sein. Sollte dies nicht der Fall sein, kontaktieren Sie bitte den FDZ-Support.

9.2 Wie kann ich mir die Pakete anzeigen lassen, die installiert sind?

Welche Pakete installiert sind und Ihnen somit für Ihre Analysen zur Verfügung stehen, können Sie sich mit dem Befehl "!pip list" anzeigen lassen. Diesen Befehl können Sie entweder in Ihrem Skript ausführen oder über die Konsole ausführen lassen:



9.3 Ich kann ein Paket nicht installieren bzw. es ist nicht in der packages Registerkarte

Durchsuchen der installierten packages

Wenn Sie die Funktionalitäten eines bestimmten packages überprüfen möchten, um zu schauen, ob es für Ihre Analysen in Frage kommt, können Sie folgenden Code am Beispiel des Paketes "tornado" nutzen:

Funktion in speziellem Package

```
import tornado
dir(tornado)  # Zeigt alle Funktionen in tornado
```

Als Output erhalten Sie folgendes Ergebnis:

```
•[6]: import tornado
      dir(tornado)  # Zeigt alle Funktionen in tornado

[6]: ['__all__',
      '__builtins__',
      '__cached__',
      '__doc__',
      '__file__',
      '__getattr__',
      '__loader__',
      '__name__',
      '__package__',
      '__path__',
      '__spec__',
      'concurrent',
      'escape',
      'gen',
      'importlib',
      'ioloop',
      'locks',
      'log',
      'platform',
      'queues',
      'speedups',
      'typing',
      'util',
      'version',
      'version_info']
```

Falls Sie nicht wissen in welchem Paket Ihre benötigte Funktion liegt, können Sie ein Skript schreiben, das **alle verfügbaren Pakete** durchgeht und prüft, ob eine Funktion existiert.

Hier ist ein entsprechendes Python-Skript am Beispiel der "Linearen Regression":

Beispielbefehl zum Durchsuchen der Pakete

```
import pkgutil
import importlib

def find_function(function_name):
    # Alle installierten Pakete ermitteln
    installed_packages = [pkg.name for pkg in pkgutil.iter_modules()]

    # Ergebnisse speichern
    found_in_packages = []

    for package in installed_packages:
        try:
            # Paket importieren
            module = importlib.import_module(package)

            # Prüfen, ob die Funktion im Modul existiert
            if hasattr(module, function_name):
                found_in_packages.append(package)

        except Exception:
            # Fehlerhafte Pakete ignorieren
            pass

    return found_in_packages

# Beispiel: Suche nach einer Funktion, z. B. "LinearRegression"
function_name = "LinearRegression"
packages_with_function = find_function(function_name)

# Ergebnisse anzeigen
if packages_with_function:
    print(f"Die Funktion '{function_name}' wurde in folgenden Paketen gefunden:")
    for package in packages_with_function:
        print(f"- {package}")
else:
    print(f"Die Funktion '{function_name}' wurde in keinem installierten Paket gefunden.")
```

```
Die Funktion 'LinearRegression' wurde in folgenden Paketen gefunden:
- statistics
```

```
[ ]:
```

9.4 Ich kann nichts in mein Git Repository pushen

Sollten Sie eine Fehlermeldung beim Pushen von Dateien in Ihr Git Repository erhalten, führen Sie zunächst den Pull Befehl aus und probieren es erneut. Sollte das Problem immer noch bestehen, notieren Sie sich am besten die Fehlermeldung und wenden sich an den FDZ-Support.

9.5 Die Datenbankverbindung in einer aktiven Session ist plötzlich unterbrochen

Aufgrund von Hintergrundprozessen kann es in seltenen Fällen passieren, dass die Datenbankverbindung neu aufgebaut wird. Daher warten Sie einige Minuten bis maximal wenigen Stunden. Sollte die Verbindung dann noch nicht wieder aufgebaut sein, kontaktieren Sie bitte den FDZ-Support.

9.5.1 Die Datenbankverbindung wird nicht von alleine aufgebaut, was kann ich tun?

Folgender Code erzeugt das CONN-Objekt um die Datenbankverbindung herzustellen:

Beispielbefehl zum nachträglichen Erstellen der Connection

```
import os
import getpass
from sqlalchemy import create_engine, Engine, Connection

username = getpass.getuser()
username = os.getenv("USER", username)
# Create the connection string
connection_string = f'hana://userkey={username}'
# Create Connection args
connection_args = {
    'encrypt': 'true',
    'sslvalidatecertificate': 'true'
}
# Create an engine
engine = create_engine(connection_string, connect_args=connection_args)
conn = engine.connect()
```

Falls statt mit sqlalchemy direkt mit dem low-level package hdbcli von SAP gearbeitet werden soll (z.Bsp. für Synthese), kann die Verbindung wie folgt hergestellt werden:

Beispielbefehl zum nachträglichen Erstellen der Connection mittels hdbcli

```
import os
import getpass
from hdbcli import dbapi

username = getpass.getuser()
username = os.getenv("USER", username)
conn = dbapi.connect(
    key=username,
    encrypt=True,
    sslValidateCertificate=True
)
cursor = conn.cursor()
```