Lab 1

In [ ]:
```
from sqlalchemy import create_engine, text
import pandas as pd

def execute(q):
    engine = create_engine('postgresql+psycopg2://vaskers5:aboba@localhost:543
    with engine.connect() as con:
        res = con.execute(text(q))
        con.commit()
        return res
```

In [ ]:
```
# Function to delete the "profi" schema
def delete_profi_schema():
    # Define the SQL query to drop the schema
    drop_schema_query = "DROP SCHEMA IF EXISTS profi CASCADE;"

    # Execute the query using the provided execute function
    execute(drop_schema_query)

# Call the function to delete the "profi" schema
delete_profi_schema()
```

In [ ]:
```
# Function to create the "profi" schema
def create_profi_schema():
    # Define the SQL query to create the schema
    create_schema_query = "CREATE SCHEMA profi;"

    # Execute the query using the provided execute function
    execute(create_schema_query)

# Call the function to create the "profi" schema
create_profi_schema()
```

In [ ]:
```


engine = create_engine('postgresql+psycopg2://vaskers5:aboba@localhost:5435/da

# Part 1: Create tables without foreign keys
q1 = """
CREATE TABLE profi.specialist (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    email VARCHAR(255),
    phone VARCHAR(50),
    address VARCHAR(255),
    city VARCHAR(255),
    passport_id VARCHAR(20),
    specialization VARCHAR(255),
    verified BOOLEAN
);
"""

q2 =  """CREATE TABLE profi.customer (
    id SERIAL PRIMARY KEY,
```

```
    name VARCHAR(255),
    email VARCHAR(255),
    phone VARCHAR(50),
    address VARCHAR(255),
    city VARCHAR(255)
);"""

q3 = """CREATE TABLE profi.review (
    id SERIAL PRIMARY KEY,
    order_id INT,
    rating INT,
    review_text TEXT,
    date TIMESTAMPTZ
);"""

q4 = """CREATE TABLE profi.order (
    id SERIAL PRIMARY KEY,
    customer_id INT,
    specialist_id INT,
    order_date TIMESTAMPTZ,
    closed BOOLEAN
);"""

q5 = """CREATE TABLE profi.category (
    id SERIAL PRIMARY KEY,
    category_name VARCHAR(255),
    description TEXT
);"""

q6 = """CREATE TABLE profi.payment (
    id SERIAL PRIMARY KEY,
    service_id INT,
    payment_date TIMESTAMPTZ,
    amount NUMERIC(10, 2),
    was_paid BOOLEAN
);
"""

q7 = """CREATE TABLE profi.service_order_table (
    id SERIAL PRIMARY KEY,
    order_id INT,
    service_id INT,
    service_price NUMERIC(10, 2),
    was_paid BOOLEAN
);
"""

q8 = """CREATE TABLE profi.service (
    id SERIAL PRIMARY KEY,
    service_name VARCHAR(255),
    category_id INT,
    description TEXT,
    price NUMERIC(10, 2)
);"""


# Execute the queries to create tables without foreign keys
for query in [q1, q2, q3, q4, q5, q6, q7, q8]:
    execute(query)
```

In [ ]:
```
# Part 2: Create foreign keys
foreign_key_queries = [
    """ALTER TABLE profi.review
    ADD FOREIGN KEY (order_id) REFERENCES profi.order(id);
    """,

    """ALTER TABLE profi.order
    ADD FOREIGN KEY (customer_id) REFERENCES profi.customer(id);

    ALTER TABLE profi.order
    ADD FOREIGN KEY (specialist_id) REFERENCES profi.specialist(id);
    """,

    """ALTER TABLE profi.payment
    ADD FOREIGN KEY (service_id) REFERENCES profi.service(id);
    """,

    """ALTER TABLE profi.service_order_table
    ADD FOREIGN KEY (order_id) REFERENCES profi.order(id);

    ALTER TABLE profi.service_order_table
    ADD FOREIGN KEY (service_id) REFERENCES profi.service(id);
    """,

    """ALTER TABLE profi.service
    ADD FOREIGN KEY (category_id) REFERENCES profi.category(id);
    """
]

# Execute the foreign key queries one by one
for query in foreign_key_queries:
    execute(query)
```

In [ ]:
```
# Execute the SQL query to get all schemas
query = """
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'profi';
"""
result = execute(query)
print(result.fetchall())
```

```
[('review',), ('service_order_table',), ('order',), ('service',), ('specialis
t',), ('category',), ('customer',), ('payment',)]
```

In [ ]:
```
import pandas as pd


df1 = pd.read_csv('specialist.csv')
df2 = pd.read_csv('user.csv')

with engine.begin() as con:
    df1.to_sql(name='specialist', con=con, if_exists='append', index=False, scl

with engine.begin() as con:
    df2.to_sql(name='customer', con=con, if_exists='append', index=False, scher
```

In [ ]:
```
df1
```

Out[ ]:

	name	email	phone	address	city	passport_id	s
0	John Doe	johndoe@example.com	(123) 555-1234	123 Main St	Cityville	1331223 12312312	
1	Jane Smith	janesmith@example.com	(456) 555-5678	456 Elm St	Townville	2334354 23456789	
2	Robert Johnson	robertjohnson@example.com	(789) 555-9012	789 Oak St	Villageton	1345345 45678901	
3	Emily Davis	emilydavis@example.com	(321) 555-3456	321 Maple St	Hamletville	2345676 56789012	
4	Michael Wilson	michaelwilson@example.com	(654) 555-7890	654 Pine St	Suburbia	3456787 67890123	
5	Jennifer Lee	jenniferlee@example.com	(987) 555-2345	987 Birch St	Countryside	4567898 78901234	
6	William Clark	williamclark@example.com	(123) 555-6789	123 Cedar St	Villageville	5678909 89012345	
7	Sarah Baker	sarahbaker@example.com	(456) 555-0123	456 Redwood St	Metroville	6789010 90123456	
8	David Lewis	davidlewis@example.com	(789) 555-3456	789 Sequoia St	Townsville	7890121 01234567	
9	Jessica Adams	jessicaadams@example.com	(321) 555-6789	321 Palm St	Cityburg	8901232 12345678	
10	James Taylor	jamestaylor@example.com	(654) 555-9012	654 Olive St	Townburg	9012343 23456789	
11	Elizabeth Martin	elizabethmartin@example.com	(987) 555-2345	987 Walnut St	Villagetown	0123454 34567890	
12	Daniel Anderson	danielanderson@example.com	(123) 555-5678	123 Pineapple St	Hamlettown	1234565 45678901	
13	Linda Hall	lindahall@example.com	(456) 555-7890	456 Banana St	Villageton	2345676 56789012	
14	Charles Harris	charlesharris@example.com	(789) 555-0123	789 Grape St	Metroburg	3456787 67890123	
15	Karen White	karenwhite@example.com	(321) 555-2345	321 Lemon St	Suburbville	4567898 78901234	
16	Matthew Moore	matthewmoore@example.com	(654) 555-	654 Strawberry	Citytown	5678909 89012345	

	name	email	phone	address	city	passport_id	s
			5678	St			
17	Patricia King	patriciaking@example.com	(987) 555-9012	987 Blueberry St	Countrysville	6789010 90123456	
18	Richard Brown	richardbrown@example.com	(123) 555-2345	123 Raspberry St	Villagecity	8901232 12345678	
19	Susan Turner	susanturner@example.com	(456) 555-6789	456 Blackberry St	Townsville	9012343 23456789	
20	Joseph Rodriguez	josephrodriguez@example.com	(789) 555-3456	789 Orange St	Metrocity	0123454 34567890	
21	Nancy Scott	nancyscott@example.com	(321) 555-9012	321 Cherry St	Hamletburg	1234565 45678901	
22	Thomas Hall	thomashall@example.com	(654) 555-2345	654 Apple St	Cityton	2345676 56789012	
23	Mary Green	marygreen@example.com	(987) 555-5678	987 Pear St	Villageton	3456787 67890123	
24	Christopher Adams	christopheradams@example.com	(123) 555-5678	123 Plum St	Countrysburg	4567898 78901234	
25	Lisa Turner	lisaturner@example.com	(456) 555-2345	456 Banana St	Suburbtown	5678909 89012345	
26	Daniel Smith	danielsmith@example.com	(789) 555-5678	789 Kiwi St	Villageburg	6789010 90123456	
27	Betty Young	bettyyoung@example.com	(321) 555-9012	321 Papaya St	Metroburg	0123454 34567890	
28	Kevin Perez	kevinperez@example.com	(654) 555-2345	654 Mango St	Cityville	1234565 45678901	
29	Dorothy Martinez	dorothymartinez@example.com	(987) 555-5678	987 Grapefruit St	Townsville	2345676 56789012	
30	Mark Johnson	markjohnson@example.com	(123) 555-9012	123 Lime St	Villageton	3456787 67890123	
31	Ashley Lee	ashleylee@example.com	(456) 555-5678	456 Lemon St	Countryside	4567898 78901234	
32	George Davis	georgedavis@example.com	(789) 555-2345	789 Orange St	Metrocity	5678909 89012345	

	name	email	phone	address	city	passport_id	s
33	Karen Taylor	karentaylor@example.com	(321) 555-9012	321 Tangerine St	Hamletburg	6789010 90123456	
34	Michael Miller	michaelmiller@example.com	(654) 555-5678	654 Clementine St	Cityton	0123454 34567890	
35	Cynthia Brown	cynthiabrown@example.com	(987) 555-5678	987 Apricot St	Villagecity	1234565 45678901	
36	Matthew Turner	matthewturner@example.com	(123) 555-2345	123 Cherry St	Metroville	2345676 56789012	
37	Donna Jackson	donnajackson@example.com	(456) 555-9012	456 Cranberry St	Townburg	3456787 67890123	
38	Paul Harris	paulharris@example.com	(789) 555-5678	789 Blueberry St	Villagetown	4567898 78901234	
39	Sharon Clark	sharonclark@example.com	(321) 555-9012	321 Raspberry St	Suburbville	5678909 89012345	
40	Edward Anderson	edwardanderson@example.com	(654) 555-5678	654 Blackberry St	Citytown	6789010 90123456	

In [ ]:
```
with engine.connect() as con:
    q = """select *
    from profi.Customer"""
    res = pd.read_sql_query(q, con)
res.head(10)
```

Out[ ]:

	id	name	email	phone	address	city
0	1	John Doe	johndoe@example.com	(123) 555-1234	123 Main St	Cityville
1	2	Jane Smith	janesmith@example.com	(456) 555-5678	456 Elm St	Townville
2	3	Robert Johnson	robertjohnson@example.com	(789) 555-9012	789 Oak St	Villageton
3	4	Emily Davis	emilydavis@example.com	(321) 555-3456	321 Maple St	Hamletville
4	5	Michael Wilson	michaelwilson@example.com	(654) 555-7890	654 Pine St	Suburbia
5	6	Jennifer Lee	jenniferlee@example.com	(987) 555-2345	987 Birch St	Countryside
6	7	William Clark	williamclark@example.com	(123) 555-6789	123 Cedar St	Villageville
7	8	Sarah Baker	sarahbaker@example.com	(456) 555-0123	456 Redwood St	Metroville
8	9	David Lewis	davidlewis@example.com	(789) 555-3456	789 Sequoia St	Townsville
9	10	Jessica Adams	jessicaadams@example.com	(321) 555-6789	321 Palm St	Cityburg

In [ ]:

```
with engine.connect() as con:
    q = """select *
    from profi.specialist"""
    res = pd.read_sql_query(q, con)
res.head(10)
```

Out[ ]:

	id	name	email	phone	address	city	passport_id	specializ
0	1	John Doe	johndoe@example.com	(123) 555-1234	123 Main St	Cityville	1331223 12312312	
1	2	Jane Smith	janesmith@example.com	(456) 555-5678	456 Elm St	Townville	2334354 23456789	inforn
2	3	Robert Johnson	robertjohnson@example.com	(789) 555-9012	789 Oak St	Villageton	1345345 45678901	cher
3	4	Emily Davis	emilydavis@example.com	(321) 555-3456	321 Maple St	Hamletville	2345676 56789012	
4	5	Michael Wilson	michaelwilson@example.com	(654) 555-7890	654 Pine St	Suburbia	3456787 67890123	inforn
5	6	Jennifer Lee	jenniferlee@example.com	(987) 555-2345	987 Birch St	Countryside	4567898 78901234	cher
6	7	William Clark	williamclark@example.com	(123) 555-6789	123 Cedar St	Villageville	5678909 89012345	
7	8	Sarah Baker	sarahbaker@example.com	(456) 555-0123	456 Redwood St	Metroville	6789010 90123456	inforn
8	9	David Lewis	davidlewis@example.com	(789) 555-3456	789 Sequoia St	Townsville	7890121 01234567	cher
9	10	Jessica Adams	jessicaadams@example.com	(321) 555-6789	321 Palm St	Cityburg	8901232 12345678	

In [ ]:

```
q = """
INSERT INTO profi.Order (customer_id, specialist_id, order_date, closed)
VALUES
    (1, 1, '2023-10-02 09:15:00', true),
    (2, 2, '2023-10-03 12:00:00', true),
    (3, 3, '2023-10-04 15:00:00', true),
    (4, 4, '2023-10-05 17:00:00', true),
    (5, 5, '2023-10-06 18:30:00', true),
    (6, 6, '2023-10-07 21:00:00', true),
    (7, 7, '2023-10-08 23:45:00', true),
    (8, 8, '2023-10-09 02:30:00', true),
    (9, 9, '2023-10-10 04:30:00', true),
    (10, 10, '2023-10-11 06:45:00', true),
    (11, 11, '2023-10-12 08:30:00', true),
    (12, 12, '2023-10-13 11:15:00', true),
    (13, 13, '2023-10-14 14:45:00', true),
    (14, 14, '2023-10-15 16:30:00', true),
    (15, 15, '2023-10-16 19:00:00', true);
"""
execute(q)
```

Out[ ]: <sqlalchemy.engine.cursor.CursorResult at 0x12e81efa0>

In [ ]:
```
with engine.connect() as con:
    q = """select *
    from profi.Order"""
    res = pd.read_sql_query(q, con)
res.head(10)
```

Out[ ]:

	id	customer_id	specialist_id	order_date	closed
0	1	1	1	2023-10-02 06:15:00+00:00	True
1	2	2	2	2023-10-03 09:00:00+00:00	True
2	3	3	3	2023-10-04 12:00:00+00:00	True
3	4	4	4	2023-10-05 14:00:00+00:00	True
4	5	5	5	2023-10-06 15:30:00+00:00	True
5	6	6	6	2023-10-07 18:00:00+00:00	True
6	7	7	7	2023-10-08 20:45:00+00:00	True
7	8	8	8	2023-10-08 23:30:00+00:00	True
8	9	9	9	2023-10-10 01:30:00+00:00	True
9	10	10	10	2023-10-11 03:45:00+00:00	True

In [ ]:
```
q = """
INSERT INTO profi.category (category_name, description)
VALUES
    ('Math', 'Mathematics is the study of numbers, quantities, and shapes.'),
    ('Informatics', 'Informatics is the science of information and computation
    ('Computer Science', 'Computer Science is the study of computers and compu
    ('History', 'The study of past events and their impact on society.'),
    ('Biology', 'The science of life and living organisms.'),
    ('Chemistry', 'The study of the composition, structure, and properties of m
    ('Physics', 'The study of the fundamental forces and properties of the univ
    ('Art', 'Expression of human creativity and imagination through various me
    ('Music', 'The art of producing sound to express emotions and ideas.'),
    ('Literature', 'Written or spoken works that convey ideas and stories.'),
    ('Geography', 'The study of the Earth and its physical features.'),
    ('Economics', 'The study of the production and distribution of goods and se
    ('Psychology', 'The science of behavior and mental processes.'),
    ('Sociology', 'The study of human society and social behavior.'),
    ('Political Science', 'The study of government and political systems.'),
    ('Philosophy', 'The exploration of fundamental questions about existence, k
    ('Environmental Science', 'The study of the environment and its impact on t
    ('Medicine', 'The science and practice of diagnosing, treating, and prevent
    ('Roof Repair', 'Professional roof repair and maintenance services.'),
    ('Apartment Renovation', 'Complete apartment renovation and remodeling.'),
    ('Plumbing Services', 'Skilled plumbing repair and installation services.'
    ('Electrical Services', 'Electrical repair, wiring, and installation.'),
    ('Landscaping', 'Landscaping design and maintenance services.'),
    ('Cleaning Services', 'Residential and commercial cleaning services.'),
    ('HVAC Services', 'Heating, ventilation, and air conditioning services.'),
    ('Pest Control', 'Pest control and extermination services.'),
    ('Auto Repair', 'Automobile repair and maintenance services.'),
    ('Interior Design', 'Interior design and decoration services.'),
```

```
    ('Legal Services', 'Legal advice and consultation services.'),
    ('Financial Services', 'Financial planning and advisory services.'),
    ('Event Planning', 'Event planning and coordination services.'),
    ('Catering Services', 'Catering and food services for events.'),
    ('Home Security', 'Home security and alarm system installation.');
"""
execute(q)
```

Out[ ]: <sqlalchemy.engine.cursor.CursorResult at 0x12e8e1580>

In [ ]:
```
with engine.connect() as con:
    q = """select *
    from profi.category"""
    res = pd.read_sql_query(q, con)
res.head(10)
```

Out[ ]:

	id	category_name	description
0	1	Math	Mathematics is the study of numbers, quantitie...
1	2	Informatics	Informatics is the science of information and ...
2	3	Computer Science	Computer Science is the study of computers and...
3	4	History	The study of past events and their impact on s...
4	5	Biology	The science of life and living organisms.
5	6	Chemistry	The study of the composition, structure, and p...
6	7	Physics	The study of the fundamental forces and proper...
7	8	Art	Expression of human creativity and imagination...
8	9	Music	The art of producing sound to express emotions...
9	10	Literature	Written or spoken works that convey ideas and ...

In [ ]:
```
review_q = """
INSERT INTO profi.Review (order_id, rating, review_text, date)
VALUES
    (1, 5, 'Отличный сервис!', '2023-10-02 09:00:00'),
    (2, 4, 'Хорошее обслуживание, но могло быть лучше.', '2023-10-03 11:30:00'
    (3, 5, 'Супер быстрая доставка!', '2023-10-04 14:45:00'),
    (4, 3, 'Опоздали с доставкой на 15 минут.', '2023-10-05 16:20:00'),
    (5, 2, 'Плохой опыт, заказ не был выполнен.', '2023-10-06 18:10:00'),
    (6, 4, 'Хорошее качество продукции.', '2023-10-07 20:05:00'),
    (7, 5, 'Очень вкусная еда!', '2023-10-08 22:30:00'),
    (8, 3, 'Среднее обслуживание.', '2023-10-09 01:15:00'),
    (9, 4, 'Приятный опыт, но цены высокие.', '2023-10-10 03:40:00'),
    (10, 1, 'Ужасное обслуживание, никогда больше не заказываю.', '2023-10-11 (
    (11, 5, 'Отличный выбор блюд.', '2023-10-12 07:55:00'),
    (12, 4, 'Меню разнообразное.', '2023-10-13 10:10:00'),
    (13, 2, 'Не рекомендую, не соответствует ожиданиям.', '2023-10-14 13:20:00
    (14, 5, 'Превосходный сервис!', '2023-10-15 15:45:00'),
    (15, 3, 'Среднее качество продукции.', '2023-10-16 18:00:00');

"""
execute(review_q)
```

Out[ ]: <sqlalchemy.engine.cursor.CursorResult at 0x12e8e7700>

In [ ]:
```
with engine.connect() as con:
    q = """select *
    from profi.Review"""
    res = pd.read_sql_query(q, con)
res.head(10)
```

Out[ ]:

	id	order_id	rating	review_text	date
0	1	1	5	Отличный сервис!	2023-10-02 06:00:00+00:00
1	2	2	4	Хорошее обслуживание, но могло быть лучше.	2023-10-03 08:30:00+00:00
2	3	3	5	Супер быстрая доставка!	2023-10-04 11:45:00+00:00
3	4	4	3	Опоздали с доставкой на 15 минут.	2023-10-05 13:20:00+00:00
4	5	5	2	Плохой опыт, заказ не был выполнен.	2023-10-06 15:10:00+00:00
5	6	6	4	Хорошее качество продукции.	2023-10-07 17:05:00+00:00
6	7	7	5	Очень вкусная еда!	2023-10-08 19:30:00+00:00
7	8	8	3	Среднее обслуживание.	2023-10-08 22:15:00+00:00
8	9	9	4	Приятный опыт, но цены высокие.	2023-10-10 00:40:00+00:00
9	10	10	1	Ужасное обслуживание, никогда больше не заказы...	2023-10-11 02:25:00+00:00

In [ ]:
```
q1 = """
INSERT INTO profi.Service (service_name, category_id, description, price)
VALUES
    ('Math Problem Solving', 1, 'Assistance with math problems and lessons', 5(
    ('Python Programming', 2, 'Python program development', 75.00),
    ('Chemical Experiments', 6, 'Conducting chemical experiments and lab work',
    ('Plumbing Installation and Repair', 23, 'Installation and repair of plumbi
    ('Music Lessons', 8, 'Individual music lessons on various instruments', 55
    ('Zoological Consultations', 4, 'Consultations on caring for domestic anima
    ('Chef Services for Events', 12, 'Conducting culinary workshops and event (
    ('Arts and Crafts', 7, 'Art and craft lessons for children and adults', 40
    ('Medical Consultations', 11, 'Medical consultations and check-ups', 90.00
    ('Home Automation', 10, 'Installation of smart home systems and automation
    ('Sports Training', 15, 'Individual and group training in various sports',
    ('Toys and Children''s Goods', 16, 'Sale and servicing of children''s toys
    ('Loyalty Programs', 13, 'Development of loyalty programs for businesses',
    ('Aviation Services', 14, 'Charter aviation flights and services', 300.00),
    ('IT Consulting', 17, 'IT consulting and information system development', 1
    ('Garden Furniture', 20, 'Sale and installation of garden furniture', 80.0(
    ('Security and Safety Services', 21, 'Security and safety services', 120.0(
"""
execute(q1)
```

Out[ ]:
```
<sqlalchemy.engine.cursor.CursorResult at 0x12e8fb280>
```

In [ ]:
```
with engine.connect() as con:
    q = """select *
    from profi.Service"""
    res = pd.read_sql_query(q, con)
res.head(10)
```

Out[ ]:

	id	service_name	category_id	description	price
0	1	Math Problem Solving	1	Assistance with math problems and lessons	50.0
1	2	Python Programming	2	Python program development	75.0
2	3	Chemical Experiments	6	Conducting chemical experiments and lab work	60.0
3	4	Plumbing Installation and Repair	23	Installation and repair of plumbing systems	70.0
4	5	Music Lessons	8	Individual music lessons on various instruments	55.0
5	6	Zoological Consultations	4	Consultations on caring for domestic animals	45.0
6	7	Chef Services for Events	12	Conducting culinary workshops and event catering	120.0
7	8	Arts and Crafts	7	Art and craft lessons for children and adults	40.0
8	9	Medical Consultations	11	Medical consultations and check-ups	90.0
9	10	Home Automation	10	Installation of smart home systems and automation	85.0

In [ ]:
```
q2 = """INSERT INTO profi.service_order_table (order_id, service_id, service_p
VALUES
    (1, 1, 50.00, true),
    (2, 2, 75.00, true),
    (3, 3, 60.00, true),
    (4, 1, 50.00, true),
    (5, 2, 75.00, true),
    (6, 3, 60.00, true),
    (7, 1, 50.00, true),
    (8, 2, 75.00, true),
    (9, 3, 60.00, true),
    (10, 1, 50.00, true),
    (11, 2, 75.00, true),
    (12, 3, 60.00, true),
    (13, 1, 50.00, true),
    (14, 2, 75.00, true),
    (15, 3, 60.00, true);
"""

execute(q2)
```

Out[ ]:
```
<sqlalchemy.engine.cursor.CursorResult at 0x12e90be20>
```

In [ ]:
```
with engine.connect() as con:
    q = """select *
    from profi.service_order_table"""
    res = pd.read_sql_query(q, con)
res.head(10)
```

Out[ ]:

	id	order_id	service_id	service_price	was_paid
0	1	1	1	50.0	True
1	2	2	2	75.0	True
2	3	3	3	60.0	True
3	4	4	1	50.0	True
4	5	5	2	75.0	True
5	6	6	3	60.0	True
6	7	7	1	50.0	True
7	8	8	2	75.0	True
8	9	9	3	60.0	True
9	10	10	1	50.0	True

In [ ]:
```
q3 = """
INSERT INTO profi.Payment (service_id, payment_date, amount, was_paid)
VALUES
    (1, '2023-10-02 09:30:00', 50.00, true),
    (2, '2023-10-03 12:15:00', 75.00, true),
    (3, '2023-10-04 15:30:00', 60.00, true),
    (4, '2023-10-05 17:45:00', 50.00, true),
    (5, '2023-10-06 19:00:00', 75.00, true),
    (6, '2023-10-07 21:15:00', 60.00, true),
    (7, '2023-10-08 23:30:00', 50.00, true),
    (8, '2023-10-09 02:45:00', 75.00, true),
    (9, '2023-10-10 04:00:00', 60.00, true),
    (10, '2023-10-11 06:15:00', 50.00, true),
    (11, '2023-10-12 08:30:00', 75.00, true),
    (12, '2023-10-13 11:45:00', 60.00, true),
    (13, '2023-10-14 14:00:00', 50.00, true),
    (14, '2023-10-15 16:15:00', 75.00, true),
    (15, '2023-10-16 18:30:00', 60.00, true),
    (3, '2022-10-04 15:30:00', 40.00, true),
    (3, '2023-10-06 15:30:00', 50.00, true)
"""
execute(q3)
```

Out[ ]:
```
<sqlalchemy.engine.cursor.CursorResult at 0x12e90bd00>
```

In [ ]:
```
with engine.connect() as con:
    q = """select *
    from profi.Payment"""
    res = pd.read_sql_query(q, con)
res.head(10)
```

Out[ ]:

	id	service_id	payment_date	amount	was_paid
0	1	1	2023-10-02 06:30:00+00:00	50.0	True
1	2	2	2023-10-03 09:15:00+00:00	75.0	True
2	3	3	2023-10-04 12:30:00+00:00	60.0	True
3	4	4	2023-10-05 14:45:00+00:00	50.0	True
4	5	5	2023-10-06 16:00:00+00:00	75.0	True
5	6	6	2023-10-07 18:15:00+00:00	60.0	True
6	7	7	2023-10-08 20:30:00+00:00	50.0	True
7	8	8	2023-10-08 23:45:00+00:00	75.0	True
8	9	9	2023-10-10 01:00:00+00:00	60.0	True
9	10	10	2023-10-11 03:15:00+00:00	50.0	True

In [ ]:
```
q = """
CREATE OR REPLACE VIEW profi.review_order_view AS
SELECT
    r.id AS review_id,
    r.order_id AS review_order_id,
    r.rating,
    r.review_text,
    r.date AS review_date,
    o.id AS order_id,
    o.customer_id,
    o.specialist_id,
    o.order_date,
    o.closed
FROM
    profi.review r
JOIN
    profi.order o ON r.order_id = o.id;
"""
execute(q)
```

Out[ ]:
```
<sqlalchemy.engine.cursor.CursorResult at 0x12e8fbfa0>
```

In [ ]:
```
with engine.connect() as con:
    q = """select *
    from profi.review_order_view"""
    res = pd.read_sql_query(q, con)
res.head(10)
```

Out[ ]:

	review_id	review_order_id	rating	review_text	review_date	order_id	customer_id	sp
0	1	1	5	Отличный сервис!	2023-10-02 06:00:00+00:00	1	1	
1	2	2	4	Хорошее обслуживание, но могло быть лучше.	2023-10-03 08:30:00+00:00	2	2	
2	3	3	5	Супер быстрая доставка!	2023-10-04 11:45:00+00:00	3	3	
3	4	4	3	Опоздали с доставкой на 15 минут.	2023-10-05 13:20:00+00:00	4	4	
4	5	5	2	Плохой опыт, заказ не был выполнен.	2023-10-06 15:10:00+00:00	5	5	
5	6	6	4	Хорошее качество продукции.	2023-10-07 17:05:00+00:00	6	6	
6	7	7	5	Очень вкусная еда!	2023-10-08 19:30:00+00:00	7	7	
7	8	8	3	Среднее обслуживание.	2023-10-08 22:15:00+00:00	8	8	
8	9	9	4	Приятный опыт, но цены высокие.	2023-10-10 00:40:00+00:00	9	9	
9	10	10	1	Ужасное обслуживание, никогда больше не заказы...	2023-10-11 02:25:00+00:00	10	10	

Lab 2

In [ ]:

```
q = """
-- Индекс на таблице "profi.payment" с использованием полей "service_id" и "pay
CREATE INDEX idx_payment_service_payment_date
ON profi.payment (service_id, payment_date);

-- Индекс на таблице "profi.service_order_table" с использованием полей "order_
CREATE INDEX idx_service_order_order_service
ON profi.service_order_table (order_id, service_id);

-- Индекс на таблице "profi.service" с использованием полей "category_id" и "pr
CREATE INDEX idx_service_category_price
ON profi.service (category_id, price);

"""
execute(q)
```

Out[ ]:

```
<sqlalchemy.engine.cursor.CursorResult at 0x10cf39fa0>
```

In [ ]:
```
q1 = """
SELECT *
FROM profi.payment
WHERE service_id = 3
AND payment_date >= '2023-01-01'
ORDER BY payment_date;"""

with engine.connect() as con:
    res = pd.read_sql_query(q1, con)
res.head(10)
```

Out[ ]:

	id	service_id	payment_date	amount	was_paid
0	3	3	2023-10-04 12:30:00+00:00	60.0	True
1	17	3	2023-10-06 12:30:00+00:00	50.0	True

In [ ]:
```
q2 = """
SELECT *
FROM profi.service_order_table
WHERE order_id = 1
AND service_id = 1
AND was_paid = true;
"""

with engine.connect() as con:
    res = pd.read_sql_query(q2, con)
res.head(10)
```

Out[ ]:

	id	order_id	service_id	service_price	was_paid
0	1	1	1	50.0	True

In [ ]:
```
q3 = """
SELECT *
FROM profi.service
WHERE category_id = 2
AND price <= 100.00;
"""

with engine.connect() as con:
    res = pd.read_sql_query(q3, con)
res.head(10)
```

Out[ ]:

	id	service_name	category_id	description	price
0	2	Python Programming	2	Python program development	75.0

In [ ]:
```
# Execute EXPLAIN for each SELECT query and display the query plans
for i, query in enumerate([q1, q2, q3]):
    print(f"Query {i+1} Plan:")
    cur = execute(f"EXPLAIN {query}")
    for plan in cur.fetchall():
        print(plan[0])
```

```
Query 1 Plan:
Sort  (cost=1.26..1.27 rows=1 width=33)
  Sort Key: payment_date
  ->  Seq Scan on payment  (cost=0.00..1.25 rows=1 width=33)
        Filter: ((payment_date >= '2023-01-01 00:00:00+03'::timestamp with tim
e zone) AND (service_id = 3))
Query 2 Plan:
Seq Scan on service_order_table  (cost=0.00..1.23 rows=1 width=29)
  Filter: (was_paid AND (order_id = 1) AND (service_id = 1))
Query 3 Plan:
Seq Scan on service  (cost=0.00..1.25 rows=1 width=572)
  Filter: ((price <= 100.00) AND (category_id = 2))
```

Lab 3

In [ ]:
```python
import hashlib

def mask_email(email):
    hash_object = hashlib.md5(email.encode())
    return hash_object.hexdigest()[:8] + '@example.com'

def mask_phone(phone):
    return '***-***-' + phone[-4:]
```

In [ ]:
```python
q1 = """
SELECT *
FROM profi.specialist
"""

q2 = """
select *
from profi.customer
"""

with engine.connect() as con:
    profi = pd.read_sql_query(q1, con)
    customers = pd.read_sql_query(q2, con)
```

In [ ]:
```python
import pandas as pd

# Assuming 'profi' is a DataFrame containing specialist and customer data
profi['email'] = profi['email'].apply(mask_email)
profi['phone'] = profi['phone'].apply(mask_phone)

customers['email'] = customers['email'].apply(mask_email)
customers['phone'] = customers['phone'].apply(mask_phone)

with engine.connect() as con:
    # Update specialist table
    profi[['id', 'email', 'phone']].to_sql(
        name='specialist_anonymized', con=con, if_exists='append', index=False
    )

    # Update customer table
    customers[['id', 'email', 'phone']].to_sql(
        name='customer_anonymized', con=con, if_exists='append', index=False, s
    )
```

In [ ]:
```
q = """
CREATE VIEW profi.full_service_order_info AS
SELECT
    sot.id AS service_order_id,
    c.id AS customer_id,
    c.name AS customer_name,
    c.email AS customer_email,
    c.phone AS customer_phone,
    c.address AS customer_address,
    sp.id AS specialist_id,
    sp.name AS specialist_name,
    sp.email AS specialist_email,
    sp.phone AS specialist_phone,
    sp.address AS specialist_address,
    sp.passport_id AS specialist_passport_id,
    sp.specialization AS specialist_specialization,
    sp.verified AS specialist_verified,
    s.id AS service_id,
    s.service_name,
    s.category_id,
    s.description AS service_description,
    s.price AS service_price,
    sot.service_price AS full_service_cost
FROM
    profi.service_order_table sot
JOIN
    profi.order o ON sot.order_id = o.id
JOIN
    profi.customer c ON o.customer_id = c.id
JOIN
    profi.specialist sp ON o.specialist_id = sp.id
JOIN
    profi.service s ON sot.service_id = s.id;
"""
execute(q)
```

Out[ ]:
```
<sqlalchemy.engine.cursor.CursorResult at 0x12efdf4c0>
```

In [ ]:
```
with engine.connect() as con:
    q = """
        select *
        from profi.full_service_order_info
    """
    full_view  = pd.read_sql_query(q, con)
full_view.head(10)
```

Out[ ]:

	service_order_id	customer_id	customer_name	customer_email	customer_phone
0	1	1	John Doe	johndoe@example.com	(123) 555-1234
1	2	2	Jane Smith	janesmith@example.com	(456) 555-5678
2	3	3	Robert Johnson	robertjohnson@example.com	(789) 555-9012
3	4	4	Emily Davis	emilydavis@example.com	(321) 555-3456
4	5	5	Michael Wilson	michaelwilson@example.com	(654) 555-7890
5	6	6	Jennifer Lee	jenniferlee@example.com	(987) 555-2345
6	7	7	William Clark	williamclark@example.com	(123) 555-6789
7	8	8	Sarah Baker	sarahbaker@example.com	(456) 555-0123
8	9	9	David Lewis	davidlewis@example.com	(789) 555-3456
9	10	10	Jessica Adams	jessicaadams@example.com	(321) 555-6789

Generalization

In [ ]:

```
q = """
CREATE MATERIALIZED VIEW profi.generalized_view AS
SELECT
    customer_id,
    AVG(service_price) AS avg_service_price,
    COUNT(service_order_id) AS order_count
FROM
    profi.full_service_order_info
GROUP BY
    customer_id;
"""
execute(q)
with engine.connect() as con:
    q = """
        select *
        from profi.generalized_view
    """
    res = pd.read_sql_query(q, con)
res.head(10)
```

Out[ ]:

	customer_id	avg_service_price	order_count
0	1	50.0	1
1	2	75.0	1
2	3	60.0	1
3	4	50.0	1
4	5	75.0	1
5	6	60.0	1
6	7	50.0	1
7	8	75.0	1
8	9	60.0	1
9	10	50.0	1

Adding Noise

In [ ]:

```
q = """
CREATE MATERIALIZED VIEW profi.noisy_view AS
SELECT
    service_order_id,
    customer_id,
    specialist_id,
    service_id,
    -- Adding noise to the order price
    service_price + (RANDOM() * 10 - 5) AS noisy_service_price
FROM
    profi.full_service_order_info;
"""
execute(q)

with engine.connect() as con:
    q = """
        select *
        from profi.noisy_view
    """
    res = pd.read_sql_query(q, con)
res.head(10)
```

Out[ ]:

	service_order_id	customer_id	specialist_id	service_id	noisy_service_price
0	1	1	1	1	53.469942
1	2	2	2	2	79.552269
2	3	3	3	3	64.270803
3	4	4	4	1	45.892369
4	5	5	5	2	79.058312
5	6	6	6	3	56.688840
6	7	7	7	1	47.105717
7	8	8	8	2	79.167532
8	9	9	9	3	62.839241
9	10	10	10	1	51.063459

Hashing

In [ ]:

```
import pandas as pd
import hashlib
from sqlalchemy import create_engine


full_view['customer_name'] = full_view['customer_name'].apply(lambda x: hashli

full_view['customer_name'] = full_view['specialist_name'].apply(lambda x: hash
# full_view = full_view.drop(columns=['customer_name', 'specialist_name'])



# Assuming you want to query the newly created table
with engine.connect() as con:
    full_view.to_sql(name='full_service_order_info', con=con, index=False, if_
    q = "SELECT * FROM full_service_order_info"
    res = pd.read_sql_query(q, con)

res.head(10)
```

Out[ ]:

	service_order_id	customer_id		customer_name
0	1	1	6cea57c2fb6cbc2a40411135005760f241fffc3e5e67ab...	jc
1	2	2	a2dd3acadb1c9dcd956216993056a7f50a9db6e3a16c60...	jan
2	3	3	c2c6ed74aea7dd7af4c54c11b806d0944e8d618184decc...	robertjc
3	4	4	6a08f0a5bae3a5dde252f2d10de649a633bcc09ca37743...	emil
4	5	5	6951b9cfb83fe1cd7659950f2a6ef246a456f06625dab1...	michael
5	6	6	54e88ee68fc97a51ef438acbe17894d2f14b10f0160c7e...	jenn
6	7	7	c39a6ccf939289cd4651f70669ef6c33e40bc5e381ce47...	willia
7	8	8	c568ada00e2e703e6c82f148b08419084acf8956e65790...	sara
8	9	9	1c1bed7fffc6294e11347dbe5705d4275faad6fd35a4b4...	davi
9	10	10	9230113d258b5894c908f13ae6e43d45788510ab4b3781...	jessica

Lab 5

1) Выберу таблицу profi.payment

2) Создам партиции на основе даты:

In [ ]:

```
q = """
CREATE TABLE profi.payment_2022 (
    CHECK (payment_date >= DATE '2022-01-01' AND payment_date < DATE '2023-01-(
) INHERITS (profi.payment);

CREATE TABLE profi.payment_2023 (
    CHECK (payment_date >= DATE '2023-01-01' AND payment_date < DATE '2024-01-(
) INHERITS (profi.payment);

CREATE TABLE profi.payment_2024 (
    CHECK (payment_date >= DATE '2024-01-01' AND payment_date < DATE '2025-01-(
) INHERITS (profi.payment);
```

```
"""
execute(q)
```

Out[ ]:
```
<sqlalchemy.engine.cursor.CursorResult at 0x12e9093a0>
```

3) Создание функции для партицирования:

In [ ]:
```
q = """

CREATE OR REPLACE FUNCTION
    insert_payment_partition()
RETURNS TRIGGER AS $$
BEGIN
IF ( NEW.payment_date >= '2022-01-01'::DATE AND NEW.payment_date < '2023-01-01
  INSERT INTO profi.payment_2022 VALUES (NEW.*);
ELSIF ( NEW.payment_date >= '2023-01-01'::DATE AND NEW.payment_date < '2024-01-
  INSERT INTO profi.payment_2023 VALUES (NEW.*);
ELSIF ( NEW.payment_date >= '2024-01-01'::DATE AND NEW.payment_date < '2025-01-
  INSERT INTO profi.payment_2024 VALUES (NEW.*);
ELSE RAISE EXCEPTION 'Date out of range. Fix the insert_payment_partition() fu
END IF;
RETURN NULL;
END;
$$
LANGUAGE plpgsql;

"""
execute(q)
```

Out[ ]:
```
<sqlalchemy.engine.cursor.CursorResult at 0x12f33de80>
```

4) Подключение функции к мастер-таблице:

In [ ]:
```
q = """
CREATE TRIGGER payment_partition_trigger
    BEFORE INSERT ON profi.payment
    FOR EACH ROW
    EXECUTE FUNCTION insert_payment_partition();

"""
execute(q)
```

Out[ ]:
```
<sqlalchemy.engine.cursor.CursorResult at 0x12f038760>
```

5) Перенос данных в партиции:

In [ ]:
```
q = """
-- Перенос данных в партиции
WITH x AS (
    DELETE FROM ONLY profi.payment
        WHERE payment_date >= '2022-01-01'::DATE and payment_date < '2023-01-01
INSERT INTO profi.payment_2022
    SELECT * FROM x;

WITH x AS (
```

```
    DELETE FROM ONLY profi.payment
        WHERE payment_date >= '2023-01-01'::DATE and payment_date < '2024-01-01
INSERT INTO profi.payment_2023
    SELECT * FROM x;

WITH x AS (
    DELETE FROM ONLY profi.payment
        WHERE payment_date >= '2024-01-01'::DATE and payment_date < '2025-01-01
INSERT INTO profi.payment_2024
    SELECT * FROM x;
"""

execute(q)
```

Out[ ]:
```
<sqlalchemy.engine.cursor.CursorResult at 0x12f349340>
```

In [ ]:
```
# Assuming you want to query the newly created table
with engine.connect() as con:
    q = "SELECT * FROM profi.payment"
    res = pd.read_sql_query(q, con)
res
```

Out[ ]:

	id	service_id	payment_date	amount	was_paid
0	16	3	2022-10-04 12:30:00+00:00	40.0	True
1	1	1	2023-10-02 06:30:00+00:00	50.0	True
2	2	2	2023-10-03 09:15:00+00:00	75.0	True
3	3	3	2023-10-04 12:30:00+00:00	60.0	True
4	4	4	2023-10-05 14:45:00+00:00	50.0	True
5	5	5	2023-10-06 16:00:00+00:00	75.0	True
6	6	6	2023-10-07 18:15:00+00:00	60.0	True
7	7	7	2023-10-08 20:30:00+00:00	50.0	True
8	8	8	2023-10-08 23:45:00+00:00	75.0	True
9	9	9	2023-10-10 01:00:00+00:00	60.0	True
10	10	10	2023-10-11 03:15:00+00:00	50.0	True
11	11	11	2023-10-12 05:30:00+00:00	75.0	True
12	12	12	2023-10-13 08:45:00+00:00	60.0	True
13	13	13	2023-10-14 11:00:00+00:00	50.0	True
14	14	14	2023-10-15 13:15:00+00:00	75.0	True
15	15	15	2023-10-16 15:30:00+00:00	60.0	True
16	17	3	2023-10-06 12:30:00+00:00	50.0	True

6) Очистка мастер-таблицы и добавление новых данных:

In [ ]:
```
q = """
-- Очистка основной таблицы
TRUNCATE ONLY profi.payment;
```

```
"""
execute(q)
```

Out[ ]:
```
<sqlalchemy.engine.cursor.CursorResult at 0x12f356ca0>
```

In [ ]:
```
q = """
INSERT INTO profi.Payment (service_id, payment_date, amount, was_paid)
VALUES
    (12, '2023-10-02 09:30:00', 50.00, false),
    (13, '2023-10-03 12:15:00', 75.00, false),
    (14, '2023-10-04 15:30:00', 60.00, false),
    (15, '2024-10-04 15:30:00', 60.00, false),
    (16, '2024-11-04 15:30:00', 60.00, false)
"""
execute(q)
```

Out[ ]:
```
<sqlalchemy.engine.cursor.CursorResult at 0x132c0e220>
```

In [ ]:
```
# Assuming you want to query the newly created table
with engine.connect() as con:
    q = "SELECT * FROM only profi.payment_2024"
    res = pd.read_sql_query(q, con)
res
```

Out[ ]:

id	service_id	payment_date	amount	was_paid

Lab6

Создание бд в монго: use profi-ru

1) Сперва для чистоты эксперимента очистим полностью схему в mongodb

In [ ]:
```
from pymongo import MongoClient

# Подключение к MongoDB
client = MongoClient('localhost', 27017)
db = client['profi-ru']  # Замените на имя базы данных в MongoDB

# Получение коллекций схемы "profi"
collections_to_clear = db.list_collection_names()
profi_collections = [collection for collection in collections_to_clear if colle

# Удаление данных из коллекций схемы "profi"
for collection_name in profi_collections:
    db[collection_name].delete_many({})
    print(f"Данные из коллекции {collection_name} удалены")
```

1)Теперь перенесем данные из posgreSQL в MongoDB, также создадим коллекции и индексы

In [ ]:
```python
from sqlalchemy import create_engine
import pandas as pd
from pymongo import MongoClient

# Подключение к PostgreSQL
engine_postgres = create_engine('postgresql+psycopg2://vaskers5:aboba@localhos

# Получение списка всех таблиц в схеме "profi"
query_tables = """
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'profi';
"""

tables = pd.read_sql_query(query_tables, engine_postgres)

# Подключение к MongoDB
client = MongoClient('localhost', 27017)
db = client['profi-ru']  # Замени на имя базы данных в MongoDB

# Перенос данных в MongoDB и создание коллекций
for table_name in tables['table_name']:
    query_data = f"SELECT * FROM profi.{table_name};"
    data = pd.read_sql_query(query_data, engine_postgres)

    # Преобразование данных в формат списка словарей для MongoDB
    data_dict = data.to_dict(orient='records')

    # Сохранение данных в коллекции MongoDB (название коллекции = имя таблицы)
    collection = db[table_name]
    collection.insert_many(data_dict)

    # Создание индексов в коллекциях MongoDB
    if table_name == 'payment':
        collection.create_index([('service_id', 1), ('payment_date', 1)])
    elif table_name == 'service_order_table':
        collection.create_index([('order_id', 1), ('service_id', 1)])
    elif table_name == 'service':
        collection.create_index([('category_id', 1), ('price', 1)])
```

2) Теперь проверим данные в таблицах

In [ ]:
```python
from sqlalchemy import create_engine
import pandas as pd
from pymongo import MongoClient

# Подключение к PostgreSQL
engine_postgres = create_engine('postgresql+psycopg2://vaskers5:aboba@localhos

# Подключение к MongoDB
client = MongoClient('localhost', 27017)
db = client['profi-ru']  # Замените на имя базы данных в MongoDB

# Проверка данных
def check_data(table_name):
    data_postgres = pd.read_sql_query(f'SELECT * FROM profi.{table_name}', eng
    data_mongodb = list(db[table_name].find())
```

```
    if data_postgres.equals(pd.DataFrame(data_mongodb).drop(["_id"], axis=1)):
        print(f"Данные для таблицы {table_name} совпадают")
    else:
        print(f"Данные для таблицы {table_name} не совпадают")

# Проверка количества записей
def check_count(table_name):
    count_postgres = pd.read_sql_query(f'SELECT COUNT(*) FROM profi.{table_name
    count_mongodb = db[table_name].count_documents({})

    if count_postgres.iloc[0, 0] == count_mongodb:
        print(f"Количество записей для таблицы {table_name} совпадает")
    else:
        print(f"Количество записей для таблицы {table_name} не совпадает")


# Таблицы для проверки
tables_to_check = ['specialist', 'customer', 'category', 'service_order_table'

for table_name in tables_to_check:
    check_data(table_name)
    check_count(table_name)
    print("__________________")
```

```
Данные для таблицы specialist совпадают
Количество записей для таблицы specialist совпадает

__________________
Данные для таблицы customer совпадают
Количество записей для таблицы customer совпадает

__________________
Данные для таблицы category совпадают
Количество записей для таблицы category совпадает

__________________
Данные для таблицы service_order_table совпадают
Количество записей для таблицы service_order_table совпадает

__________________
Данные для таблицы service совпадают
Количество записей для таблицы service совпадает

__________________
```

3) Пытливый ум заметит, что я проверил не все таблицы - это связано с тем, что формат хранения даты отличается, вот ниже пруф

In [ ]:
```
table_name = "review"
data_postgres = pd.read_sql_query(f'SELECT * FROM profi.{table_name}', engine_
data_mongodb = pd.DataFrame(list(db[table_name].find())).drop(["_id"], axis=1)

if data_postgres.equals(data_mongodb):
    print(f"Данные для таблицы {table_name} совпадают")
else:
    print(f"Данные для таблицы {table_name} не совпадают")
```

```
Данные для таблицы review не совпадают
```

In [ ]:
```
data_postgres
```

Out[ ]:

	id	order_id	rating	review_text	date
0	1	1	5	Отличный сервис!	2023-10-02 06:00:00+00:00
1	2	2	4	Хорошее обслуживание, но могло быть лучше.	2023-10-03 08:30:00+00:00
2	3	3	5	Супер быстрая доставка!	2023-10-04 11:45:00+00:00
3	4	4	3	Опоздали с доставкой на 15 минут.	2023-10-05 13:20:00+00:00
4	5	5	2	Плохой опыт, заказ не был выполнен.	2023-10-06 15:10:00+00:00
5	6	6	4	Хорошее качество продукции.	2023-10-07 17:05:00+00:00
6	7	7	5	Очень вкусная еда!	2023-10-08 19:30:00+00:00
7	8	8	3	Среднее обслуживание.	2023-10-08 22:15:00+00:00
8	9	9	4	Приятный опыт, но цены высокие.	2023-10-10 00:40:00+00:00
9	10	10	1	Ужасное обслуживание, никогда больше не заказы...	2023-10-11 02:25:00+00:00
10	11	11	5	Отличный выбор блюд.	2023-10-12 04:55:00+00:00
11	12	12	4	Меню разнообразное.	2023-10-13 07:10:00+00:00
12	13	13	2	Не рекомендую, не соответствует ожиданиям.	2023-10-14 10:20:00+00:00
13	14	14	5	Превосходный сервис!	2023-10-15 12:45:00+00:00
14	15	15	3	Среднее качество продукции.	2023-10-16 15:00:00+00:00

In [ ]:
```
data_mongodb
```

Out[ ]:

	id	order_id	rating	review_text	date
0	1	1	5	Отличный сервис!	2023-10-02 06:00:00
1	2	2	4	Хорошее обслуживание, но могло быть лучше.	2023-10-03 08:30:00
2	3	3	5	Супер быстрая доставка!	2023-10-04 11:45:00
3	4	4	3	Опоздали с доставкой на 15 минут.	2023-10-05 13:20:00
4	5	5	2	Плохой опыт, заказ не был выполнен.	2023-10-06 15:10:00
5	6	6	4	Хорошее качество продукции.	2023-10-07 17:05:00
6	7	7	5	Очень вкусная еда!	2023-10-08 19:30:00
7	8	8	3	Среднее обслуживание.	2023-10-08 22:15:00
8	9	9	4	Приятный опыт, но цены высокие.	2023-10-10 00:40:00
9	10	10	1	Ужасное обслуживание, никогда больше не заказы...	2023-10-11 02:25:00
10	11	11	5	Отличный выбор блюд.	2023-10-12 04:55:00
11	12	12	4	Меню разнообразное.	2023-10-13 07:10:00
12	13	13	2	Не рекомендую, не соответствует ожиданиям.	2023-10-14 10:20:00
13	14	14	5	Превосходный сервис!	2023-10-15 12:45:00
14	15	15	3	Среднее качество продукции.	2023-10-16 15:00:00

4) Теперь создадим представления

In [ ]:

```
import subprocess

# Пример команды Mongo Shell
mongo_commands = """
use profi-ru
db.createView(
   "specialist_order_view",
   "order",
   [
     { $lookup:
        {
          from: "specialist",
          localField: "specialist_id",
          foreignField: "id",
          as: "specialist"
```

```
        }
      },
      { $unwind: "$specialist" },
      { $project: { _id: 0, order_id: "$id", specialist_name: "$specialist.name"
    ]
)
"""

# Запуск команды в Mongo Shell из Python
process = subprocess.Popen(["mongosh"], stdin=subprocess.PIPE, stdout=subproces
stdout, stderr = process.communicate(input=mongo_commands)

# Запрос данных из представления и преобразование в DataFrame через Pandas
result = db.specialist_order_view.find({})  # Запрос данных из представления
df = pd.DataFrame(list(result))  # Преобразование результатов запроса в DataFra

# Вывод данных для проверки
df
```

Out[ ]:

	order_date	order_id	specialist_name
0	2023-10-02 06:15:00	1	John Doe
1	2023-10-03 09:00:00	2	Jane Smith
2	2023-10-04 12:00:00	3	Robert Johnson
3	2023-10-05 14:00:00	4	Emily Davis
4	2023-10-06 15:30:00	5	Michael Wilson
5	2023-10-07 18:00:00	6	Jennifer Lee
6	2023-10-08 20:45:00	7	William Clark
7	2023-10-08 23:30:00	8	Sarah Baker
8	2023-10-10 01:30:00	9	David Lewis
9	2023-10-11 03:45:00	10	Jessica Adams
10	2023-10-12 05:30:00	11	James Taylor
11	2023-10-13 08:15:00	12	Elizabeth Martin
12	2023-10-14 11:45:00	13	Daniel Anderson
13	2023-10-15 13:30:00	14	Linda Hall
14	2023-10-16 16:00:00	15	Charles Harris

In [ ]:

```
# Создание представления service_order_view
mongo_commands = """
use profi-ru
db.createView(
   "service_order_view",
   "service_order_table",
   [
     { $lookup:
        {
          from: "service",
          localField: "service_id",
          foreignField: "id",
          as: "service"
```

```
        }
      },
      { $unwind: "$service" },
      { $lookup:
         {
           from: "order",
           localField: "order_id",
           foreignField: "id",
           as: "order"
         }
      },
      { $unwind: "$order" },
      { $project: { _id: 0, order_id: "$order.id", service_name: "$service.servi
   ]
)
"""

# Запуск команды в Mongo Shell из Python
process = subprocess.Popen(["mongosh"], stdin=subprocess.PIPE, stdout=subproces
stdout, stderr = process.communicate(input=mongo_commands)

# Запрос данных из представления и преобразование в DataFrame через Pandas
result_service = db.service_order_view.find({})  # Запрос данных из представле
df_service = pd.DataFrame(list(result_service))  # Преобразование результатов

# Вывод данных для проверки
df_service
```

Out[ ]:

	was_paid	order_id	service_name	service_price
0	True	1	Math Problem Solving	50.0
1	True	2	Python Programming	75.0
2	True	3	Chemical Experiments	60.0
3	True	4	Math Problem Solving	50.0
4	True	5	Python Programming	75.0
5	True	6	Chemical Experiments	60.0
6	True	7	Math Problem Solving	50.0
7	True	8	Python Programming	75.0
8	True	9	Chemical Experiments	60.0
9	True	10	Math Problem Solving	50.0
10	True	11	Python Programming	75.0
11	True	12	Chemical Experiments	60.0
12	True	13	Math Problem Solving	50.0
13	True	14	Python Programming	75.0
14	True	15	Chemical Experiments	60.0