TECHNOLOGIES DATABASE

FINAL PROJECT FOR ADVANCED DATABASE MANAGEMENTSYSTEM-GROUP3

USF UNIVERSITY OF SOUTH FLORIDA.

Team Members

Subash Chandra Biswal

Sai Kiran Batchu

Venkata Naga Sukumar Vinnakota

Kushal Reddy Chavva

Naresh Goud Aakula

Table of Contents

List of Figures

Topic Area	Description	Group Members	Weight
Database Design	This part should include a logical database design (for the relational model), using normalization to control redundancy and integrity constraints for data quality.	Subash Chandra Biswal Sai Kiran Batchu Venkata Naga Sukumar Vinnakota Kushal Reddy Chavva Naresh Goud Aakula	20%
Query Writing	This part is another chance to write SQL queries, explore transactions, and even do some database programming for stored procedures.	Subash Chandra Biswal Sai Kiran Batchu Venkata Naga Sukumar Vinnakota Kushal Reddy Chavva	30%
Performance Tuning	In this section, you can capitalize and extend your prior experiments with indexing, optimizer modes, partitioning, parallel execution, and any other techniques you want to further explore.	Subash Chandra Biswal Sai Kiran Batchu Venkata Naga Sukumar Vinnakota Kushal Reddy Chavva	20%
DBA Scripts & Data Visualization	Here you are free to explore any other topics of interest. Suggestions include DBA scripts, database security, interface design, data visualization, data mining, and NoSQL databases.	Subash Chandra Biswal Sai Kiran Batchu Venkata Naga Sukumar Vinnakota Kushal Reddy Chavva	30%

PURPOSE

The purpose of this document is to explain in detail the design process involved in creating and designing a database for the technologies. This document describes all the attributes and entities involved in the database. The end goal is to design a database for all available technologies in the world in such a way that there is seamless access to any required form of data for a particular technology/certification/learning or any business use case, no matter how complex, to ensure the fastest possible data retrieval & instant insights from the available data leveraging preloaded intelligent analytics & data visualization. The database will be used to develop an application to show all available technologies, their parent or client companies. It will allow to register customers, allow them to subscribe for technology or learning, and register for certification. For each technology, it will show the technology stack it comes under and all available learning and certification options. Also, it shows the use cases of each technology, the rank of technology in the technology stack, and any prerequisite technology that needs to be learned before learning it.

NARRATIVE

As part of this group project, we identified a potential issue we face when we try to find details about technology or tool. This issue is mostly faced by students, IT professionals, or instructors. To know about a specific technology, we usually google search or go to the technology's website to know details like what the technology is about, what is its use case, and the types of subscription options available. To know about the price of any subscription, we should refer to the technology's website. Also, to know about the learning options for a technology we will get thousands of paid or free options which we have to filter out and then keep track of all the favorable resources to decide on one. Another part is the certification where, to get the details about the available certifications for technology, we should follow the conventional method of search engine and if we miss a track of something we have to redo the whole process.

To solve this problem, we have planned to propose a database for information about all the technologies or tools available in the market. The broader idea is to launch a website that will use this database and make the information available in one place. It will have all the free materials ready to download and hyperlinks for all the paid services. It will be a single source to go for any technology related queries.

Now, we have only covered the design for master tables which will store all the information about technologies, owner and client companies, technology stacks, use cases, certifications, subscription options, customer details, and addresses.

There is scope to extend this database to include transaction, analytical, and reporting data.

We came up with an Entity Relationship Diagram depicted in the section 'Entity-Relationship Diagram Representing Database Design'. The ER diagram has the attributes of entities and relations between different entities. Entities will be considered as tables, attributes as columns and tuples as rows during implementation of the project.

The database is normalized up to 3rd normal form.

ENTITIES

1. TECHNOLGY
2. TECHNOLOGY_TYPE
3. TECHNOLOGY_STACK
4. TECH_STACK_MAP
5. OWNER_COMPANIES
6. CLIENT_COMPANIES
7. TECH_COMP_MAP
8. CUSTOMER
9. COUNTRY_ADDRESS
10. LOCAL_ADDRESS
11. CERTIFICATION
12. LEARNING
13. SUBSCRIPTION
14. USE_CASES

ENTITIES WITH ATTRIBUTES

Entity Name	Attributes
TECHNOLOGY	➢ TECH_ID ➢ NAME ➢ LAUNCH_DATE ➢ IS_OPEN_SOURCE ➢ DOWNLOAD_SITE
TECHNOLOGY_TYPE	➢ TYPE_ID ➢ TECH_ID ➢ TYPE
TECHNOLOGY_STACK	➢ STACK_ID ➢ STACK_NAME ➢ STACK_SUB_ID
TECH_STACK_MAP	➢ TECH_ID ➢ STACK_ID
OWNER_COMPANIES	➢ O_COMP_ID ➢ TECH_ID ➢ COUNTRY_ADD_ID ➢ NAME
CLIENT_COMPANIES	➢ COMP_ID ➢ COUNTRY_ADD_ID ➢ COMP_NAME ➢ BUSINESS_TYPE
TECH_COMP_MAP	➢ TECH_ID ➢ COMP_ID
CUSTOMERS	➢ CUST_ID ➢ INITIAL_NAME ➢ FIRST_NAME ➢ MIDDLE_NAME ➢ LAST_NAME
COUNTRY_ADDRESS	➢ COUNTRY_ADD_ID ➢ COUNTRY ➢ STATE ➢ CITY ➢ AREA_CODE ➢ ZIPCODE
LOCAL_ADDRESS	➢ LOCAL_ADD_ID ➢ CUST_ID ➢ COUNTRY_ADD_ID ➢ STREET_NAME ➢ APARTMENT_NUM ➢ OFFICE_NUM

	➢ BUILDING_NUM ➢ BUILDING_NAME ➢ LANDMARK ➢ PHONE ➢ ISD_CODE ➢ IS_CURRENT_ADDRESS
CERTIFICATION	➢ CERT_ID ➢ TECH_ID ➢ CERT_CODE ➢ CERT_NAME ➢ CERT_PATH_ID ➢ CERT_SEQ_IN_PATH ➢ REGISTRATION_URL ➢ PRICE ➢ CURRENCY ➢ EXAM_MODE ➢ EXAM_DURATION ➢ DURATION_UNIT
LEARNING	➢ LEARN_ID ➢ TECH_ID ➢ LEARN_MODE ➢ REG_URL ➢ IS_FREE ➢ PRICE
SUBSCRIPTION	➢ SUBSCRIPTION_ID ➢ TECH_ID ➢ SUB_TYPE ➢ IS_FREE ➢ SUB_NAME ➢ DOWNLOAD_URL ➢ PREREQUISITE
USE_CASES	➢ USE_ID ➢ TECH_ID ➢ USE_NAME ➢ SCENARIO_RANK ➢ PRIOR_KNOWLEDGE

BUSINESS RULES

- A technology stack can have many technologies and each technology can belong to many technology stacks.
- A technology can belong to many technology types.
- A technology can have many use cases.
- A technology can be owned by many companies.
- A technology can have many client companies and a client company can use many technologies.
- A customer can subscribe, learn, or register for certification for many technologies.
- The local address of only customers is maintained, and technology will have only a country-specific address.
- Any subscription or learning or certification can be free or paid.
- Local address keeps a history of addresses of any customer.

ENTITY-RELATIONSHIP DIAGRAM REPRESENTING DATABASE DESIGN

Proposed database design

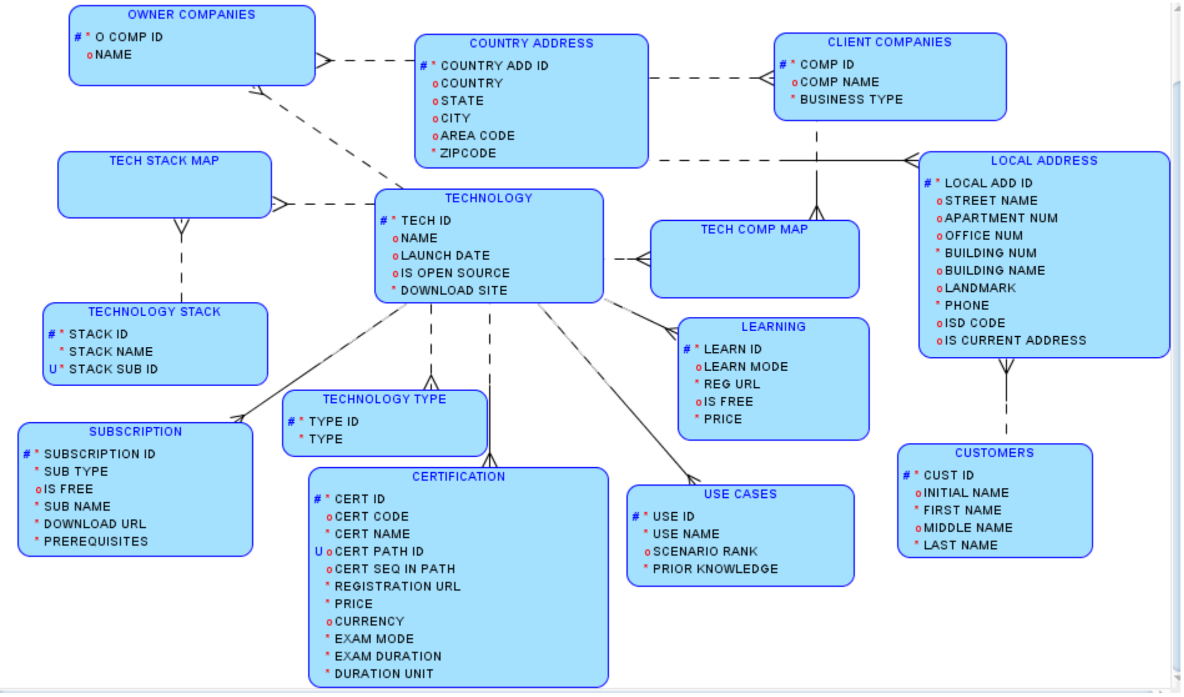


Figure 1: Proposed Database Design

Implemented design in Oracle SQL-

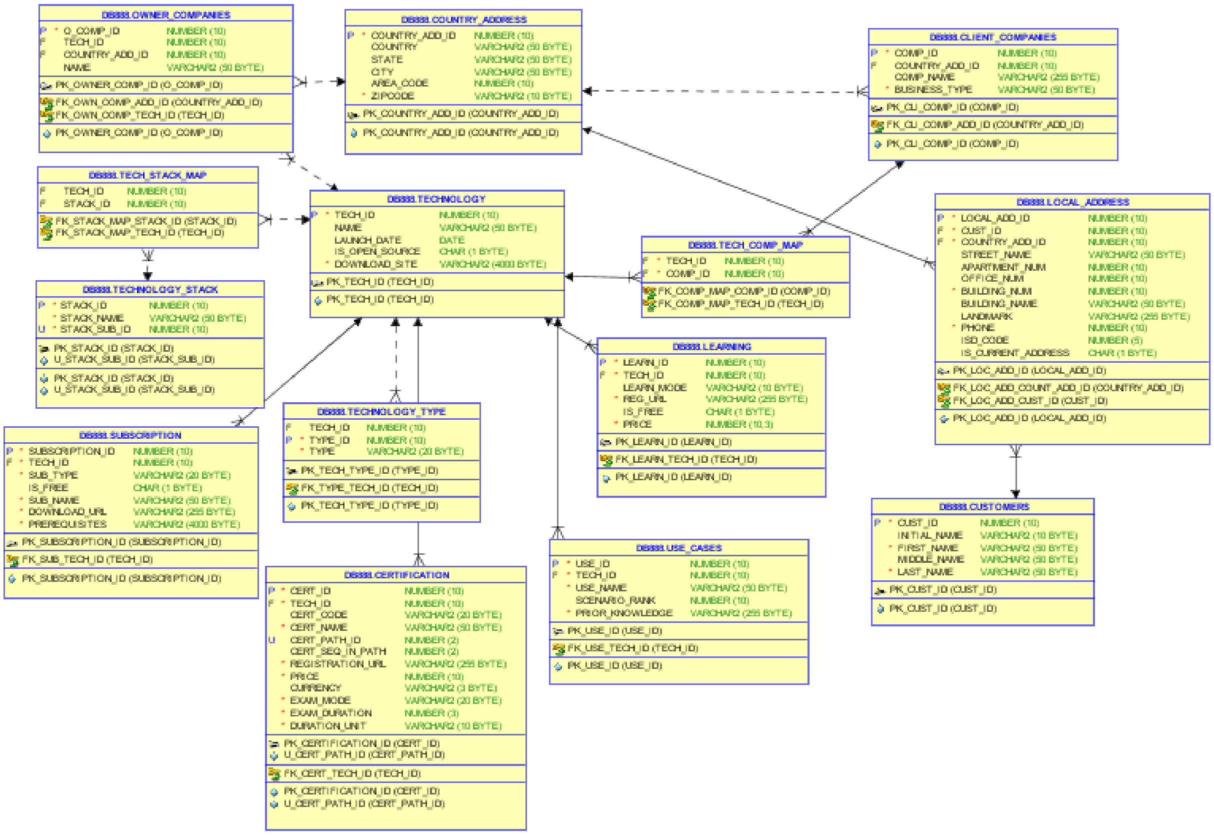
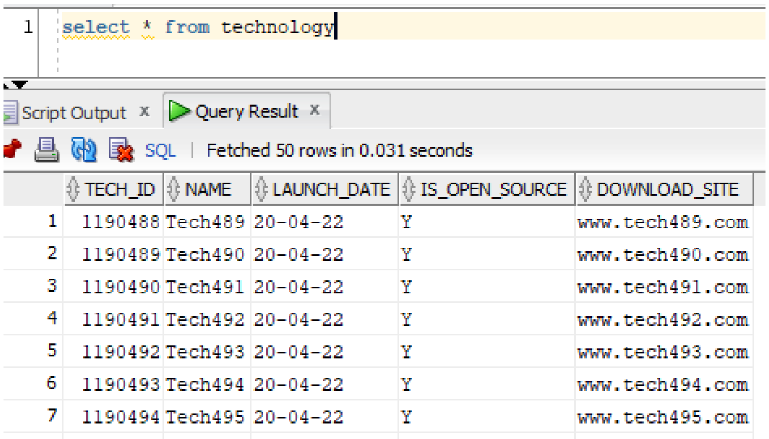

DB888.OWNER_COMPANIES
P * O_COMP_ID NUMBER (10)
F TECH_ID NUMBER (10)
F COUNTRY_ADD_ID NUMBER (10)
NAME VARCHAR2 (50 BYTE)
PK_OWNER_COMP_ID (O_COMP_ID)
FK_OWN_COMP_ADD_ID (COUNTRY_ADD_ID)
FK_OWN_COMP_TECH_ID (TECH_ID)
PK_OWNER_COMP_ID (O_COMP_ID)
DB888.COUNTRY_ADDRESS
P * COUNTRY_ADD_ID NUMBER (10)
COUNTRY VARCHAR2 (50 BYTE)
STATE VARCHAR2 (50 BYTE)
CITY VARCHAR2 (50 BYTE)
AREA_CODE NUMBER (10)
* ZIPCODE VARCHAR2 (10 BYTE)
PK_COUNTRY_ADD_ID (COUNTRY_ADD_ID)
PK_COUNTRY_ADD_ID (COUNTRY_ADD_ID)
DB888.CLIENT_COMPANIES
P * COMP_ID NUMBER (10)
F COUNTRY_ADD_ID NUMBER (10)
COMP_NAME VARCHAR2 (255 BYTE)
* BUSINESS_TYPE VARCHAR2 (50 BYTE)
PK_CLI_COMP_ID (COMP_ID)
FK_CLI_COMP_ADD_ID (COUNTRY_ADD_ID)
PK_CLI_COMP_ID (COMP_ID)
DB888.TECH_STACK_MAP
F TECH_ID NUMBER (10)
F STACK_ID NUMBER (10)
FK_STACK_MAP_STACK_ID (STACK_ID)
FK_STACK_MAP_TECH_ID (TECH_ID)
DB888.TECHNOLOGY
P * TECH_ID NUMBER (10)
NAME VARCHAR2 (50 BYTE)
LAUNCH_DATE DATE
IS_OPEN_SOURCE CHAR (1 BYTE)
* DOWNLOAD_SITE VARCHAR2 (4000 BYTE)
PK_TECH_ID (TECH_ID)
PK_TECH_ID (TECH_ID)
DB888.TECH_COMP_MAP
F * TECH_ID NUMBER (10)
F * COMP_ID NUMBER (10)
FK_COMP_MAP_COMP_ID (COMP_ID)
FK_COMP_MAP_TECH_ID (TECH_ID)
DB888.LOCAL_ADDRESS
P * LOCAL_ADD_ID NUMBER (10)
F * CUST_ID NUMBER (10)
F * COUNTRY_ADD_ID NUMBER (10)
STREET_NAME VARCHAR2 (50 BYTE)
APARTMENT_NUM NUMBER (10)
OFFICE_NUM NUMBER (10)
* BUILDING_NUM NUMBER (10)
BUILDING_NAME VARCHAR2 (50 BYTE)
LANDMARK VARCHAR2 (255 BYTE)
* PHONE NUMBER (10)
ISD_CODE NUMBER (5)
IS_CURRENT_ADDRESS CHAR (1 BYTE)
PK_LOC_ADD_ID (LOCAL_ADD_ID)
FK_LOC_ADD_COUNT_ADD_ID (COUNTRY_ADD_ID)
FK_LOC_ADD_CUST_ID (CUST_ID)
PK_LOC_ADD_ID (LOCAL_ADD_ID)
DB888.TECHNOLOGY_STACK
P * STACK_ID NUMBER (10)
* STACK_NAME VARCHAR2 (50 BYTE)
U * STACK_SUB_ID NUMBER (10)
PK_STACK_ID (STACK_ID)
U_STACK_SUB_ID (STACK_SUB_ID)
PK_STACK_ID (STACK_ID)
U_STACK_SUB_ID (STACK_SUB_ID)
DB888.LEARNING
P * LEARN_ID NUMBER (10)
F * TECH_ID NUMBER (10)
LEARN_MODE VARCHAR2 (10 BYTE)
* REG_URL VARCHAR2 (255 BYTE)
IS_FREE CHAR (1 BYTE)
* PRICE NUMBER (10,3)
PK_LEARN_ID (LEARN_ID)
FK_LEARN_TECH_ID (TECH_ID)
PK_LEARN_ID (LEARN_ID)
DB888.TECHNOLOGY_TYPE
F TECH_ID NUMBER (10)
P * TYPE_ID NUMBER (10)
* TYPE VARCHAR2 (20 BYTE)
PK_TECH_TYPE_ID (TYPE_ID)
FK_TYPE_TECH_ID (TECH_ID)
PK_TECH_TYPE_ID (TYPE_ID)
DB888.SUBSCRIPTION
P * SUBSCRIPTION_ID NUMBER (10)
F * TECH_ID NUMBER (10)
* SUB_TYPE VARCHAR2 (20 BYTE)
IS_FREE CHAR (1 BYTE)
* SUB_NAME VARCHAR2 (50 BYTE)
* DOWNLOAD_URL VARCHAR2 (255 BYTE)
* PREREQUISITES VARCHAR2 (4000 BYTE)
PK_SUBSCRIPTION_ID (SUBSCRIPTION_ID)
FK_SUB_TECH_ID (TECH_ID)
PK_SUBSCRIPTION_ID (SUBSCRIPTION_ID)
DB888.CUSTOMERS
P * CUST_ID NUMBER (10)
INITIAL_NAME VARCHAR2 (10 BYTE)
* FIRST_NAME VARCHAR2 (50 BYTE)
MIDDLE_NAME VARCHAR2 (50 BYTE)
* LAST_NAME VARCHAR2 (50 BYTE)
PK_CUST_ID (CUST_ID)
PK_CUST_ID (CUST_ID)
DB888.USE_CASES
P * USE_ID NUMBER (10)
F * TECH_ID NUMBER (10)
* USE_NAME VARCHAR2 (50 BYTE)
SCENARIO_RANK NUMBER (10)
* PRIOR_KNOWLEDGE VARCHAR2 (255 BYTE)
PK_USE_ID (USE_ID)
FK_USE_TECH_ID (TECH_ID)
PK_USE_ID (USE_ID)
DB888.CERTIFICATION
P * CERT_ID NUMBER (10)
F * TECH_ID NUMBER (10)
CERT_CODE VARCHAR2 (20 BYTE)
* CERT_NAME VARCHAR2 (50 BYTE)
U CERT_PATH_ID NUMBER (2)
CERT_SEQ_IN_PATH NUMBER (2)
* REGISTRATION_URL VARCHAR2 (255 BYTE)
* PRICE NUMBER (10)
CURRENCY VARCHAR2 (3 BYTE)
* EXAM_MODE VARCHAR2 (20 BYTE)
* EXAM_DURATION NUMBER (3)
* DURATION_UNIT VARCHAR2 (10 BYTE)
PK_CERTIFICATION_ID (CERT_ID)
U_CERT_PATH_ID (CERT_PATH_ID)
FK_CERT_TECH_ID (TECH_ID)
PK_CERTIFICATION_ID (CERT_ID)
U_CERT_PATH_ID (CERT_PATH_ID)

Figure 2: Implemented design in Oracle SQL

Note: This database is to store all master data only. The transaction table will be developed in the future.

TABLE VIEWS

1. **Technology:** This table keeps all details about each technology such as Technology ID(primary key), name, launch date, free or paid technology, and the URL for downloading the technology.



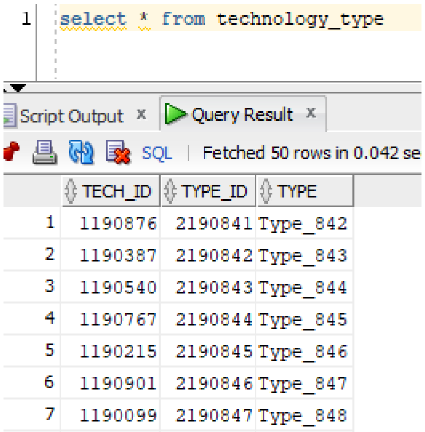
```
select * from technology
```

Fetched 50 rows in 0.031 seconds

	TECH_ID	NAME	LAUNCH_DATE	IS_OPEN_SOURCE	DOWNLOAD_SITE
1	1190488	Tech489	20-04-22	Y	www.tech489.com
2	1190489	Tech490	20-04-22	Y	www.tech490.com
3	1190490	Tech491	20-04-22	Y	www.tech491.com
4	1190491	Tech492	20-04-22	Y	www.tech492.com
5	1190492	Tech493	20-04-22	Y	www.tech493.com
6	1190493	Tech494	20-04-22	Y	www.tech494.com
7	1190494	Tech495	20-04-22	Y	www.tech495.com

Figure 3: Technology Table

2. **Technology_type**: This table contains the details of the type of each technology such as Type ID (Primary Key), Tech ID(Foreign Key), and type (Database, Programming Language, Operating System, etc).



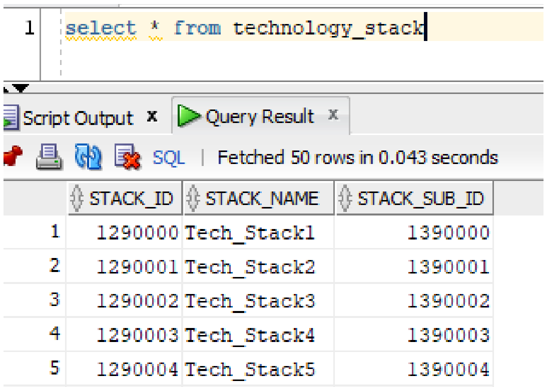
```
select * from technology_type
```

Fetched 50 rows in 0.042 se

	TECH_ID	TYPE_ID	TYPE
1	1190876	2190841	Type_842
2	1190387	2190842	Type_843
3	1190540	2190843	Type_844
4	1190767	2190844	Type_845
5	1190215	2190845	Type_846
6	1190901	2190846	Type_847
7	1190099	2190847	Type_848

Figure 4: Technology Type Table

3. **Technology_stack:** This table contains the details of tech stack for each technology. It has attributes like Stack_ ID (Primary Key), Tech ID (Foreign Key), stack sub-id, and stack names such as (Cloud, Web, etc.).



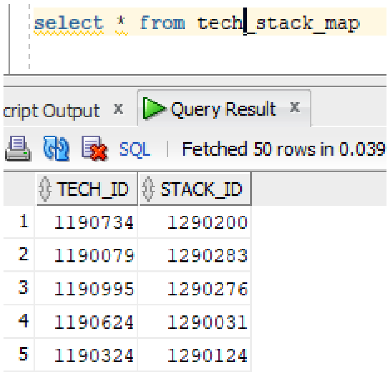
```
select * from technology_stack
```

Fetched 50 rows in 0.043 seconds

	STACK_ID	STACK_NAME	STACK_SUB_ID
1	1290000	Tech_Stack1	1390000
2	1290001	Tech_Stack2	1390001
3	1290002	Tech_Stack3	1390002
4	1290003	Tech_Stack4	1390003
5	1290004	Tech_Stack5	1390004

Figure 5: Technology Stack Table

4. **Tech_stack_map:** This table is the intermediate table that maps the Technology table and the Technology_stack table based on attributed Tech ID and Stack ID.



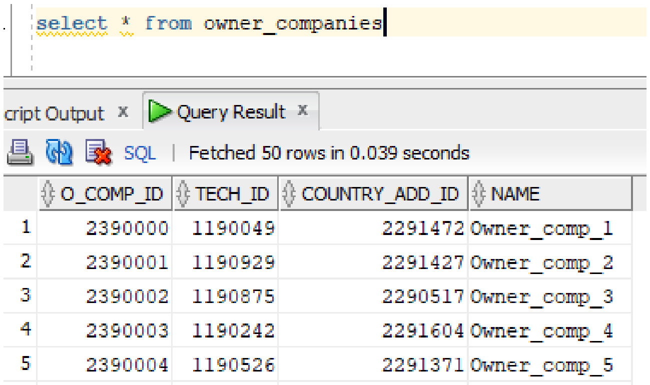
```
select * from tech_stack_map
```

Fetched 50 rows in 0.039

	TECH_ID	STACK_ID
1	1190734	1290200
2	1190079	1290283
3	1190995	1290276
4	1190624	1290031
5	1190324	1290124

Figure 6: Tech Stack Map Table

5. **Owner_companies**: This table keeps data about the owner companies of the technologies such as Owner Comp ID (primary key), Tech ID (Foreign key), Country Address ID (Foreign Key), Company Name, etc.



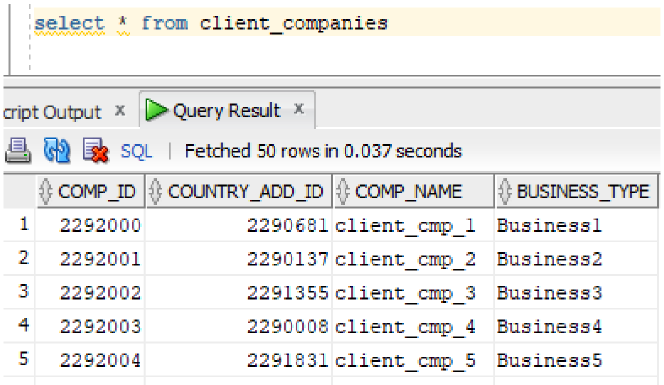
```
select * from owner_companies
```

Fetched 50 rows in 0.039 seconds

	O_COMP_ID	TECH_ID	COUNTRY_ADD_ID	NAME
1	2390000	1190049	2291472	Owner_comp_1
2	2390001	1190929	2291427	Owner_comp_2
3	2390002	1190875	2290517	Owner_comp_3
4	2390003	1190242	2291604	Owner_comp_4
5	2390004	1190526	2291371	Owner_comp_5

Figure 7: Owner Companies Table

6. **Client_Companies:** This table keeps data about the client companies of the technologies such as Comp ID (primary key), Country Address ID (Foreign Key), Company Name, Business Type, etc.



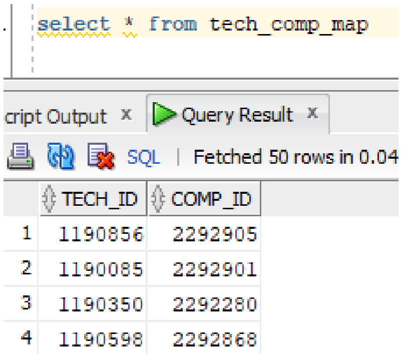
```
select * from client_companies
```

Fetched 50 rows in 0.037 seconds

	COMP_ID	COUNTRY_ADD_ID	COMP_NAME	BUSINESS_TYPE
1	2292000	2290681	client_cmp_1	Business1
2	2292001	2290137	client_cmp_2	Business2
3	2292002	2291355	client_cmp_3	Business3
4	2292003	2290008	client_cmp_4	Business4
5	2292004	2291831	client_cmp_5	Business5

Figure 8: Client Companies Table

7. **Tech_comp_map:** This table is the intermediate table that maps Technology table and Client_Companies table based on attributed Tech ID and Comp ID.



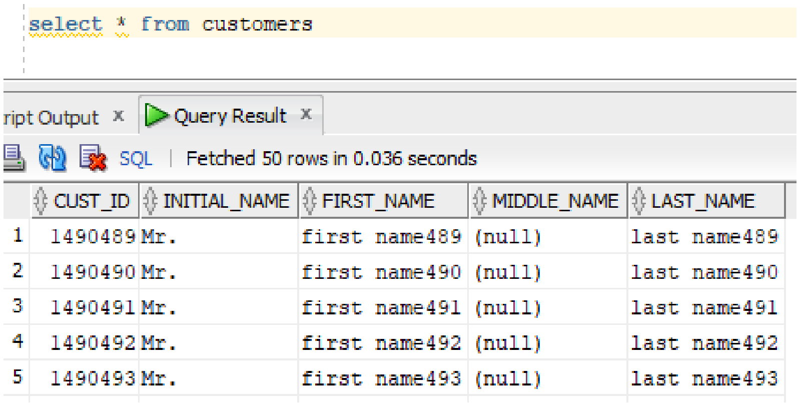
```
select * from tech_comp_map
```

Fetched 50 rows in 0.04

	TECH_ID	COMP_ID
1	1190856	2292905
2	1190085	2292901
3	1190350	2292280
4	1190598	2292868

Figure 9: Tech Comp Map Table

8. **Customers:** This table keeps data about all customers such as Customer ID(primary key), initial name, first name, middle name, and last name.



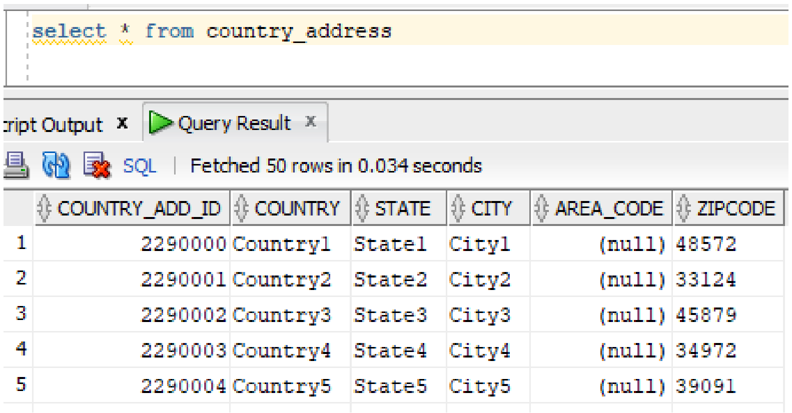
```
select * from customers
```

Fetched 50 rows in 0.036 seconds

	CUST_ID	INITIAL_NAME	FIRST_NAME	MIDDLE_NAME	LAST_NAME
1	1490489	Mr.	first name489	(null)	last name489
2	1490490	Mr.	first name490	(null)	last name490
3	1490491	Mr.	first name491	(null)	last name491
4	1490492	Mr.	first name492	(null)	last name492
5	1490493	Mr.	first name493	(null)	last name493

Figure 10: Customers Table

9. **Country_Address:** This table keeps data about the country-specific address details such as Country Address ID (Primary Key), country name, state name, city name, area code, and Zipcode.



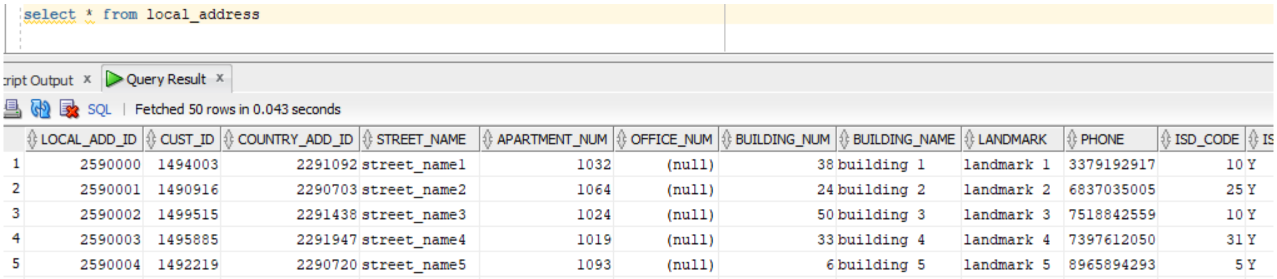
```
select * from country_address
```

Fetched 50 rows in 0.034 seconds

	COUNTRY_ADD_ID	COUNTRY	STATE	CITY	AREA_CODE	ZIPCODE
1	2290000	Country1	State1	City1	(null)	48572
2	2290001	Country2	State2	City2	(null)	33124
3	2290002	Country3	State3	City3	(null)	45879
4	2290003	Country4	State4	City4	(null)	34972
5	2290004	Country5	State5	City5	(null)	39091

Figure 11: County Address Table

10. **Local_Address:** This table keeps the addresses of all customers such as Local Address ID (Primary Key), Customer ID (Foreign Key), Country Address ID (Foreign Key), street name, apartment number, office number, building number, building name, landmark, phone, ISD code and whether the address is recent one or older (Y/N).



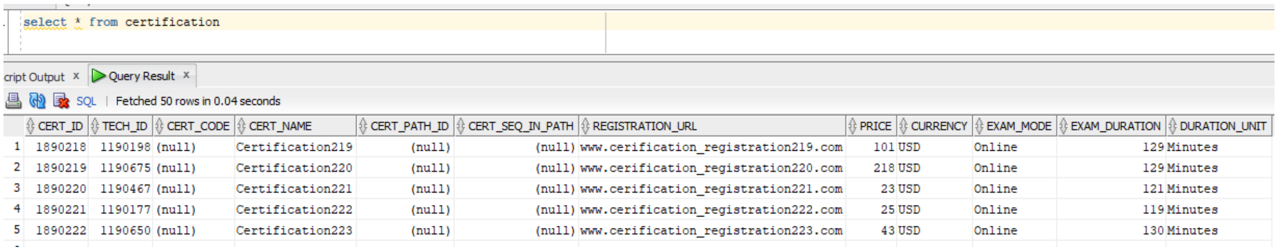
```
select * from local_address
```

Fetched 50 rows in 0.043 seconds

	LOCAL_ADD_ID	CUST_ID	COUNTRY_ADD_ID	STREET_NAME	APARTMENT_NUM	OFFICE_NUM	BUILDING_NUM	BUILDING_NAME	LANDMARK	PHONE	ISD_CODE	IS
1	2590000	1494003	2291092	street_name1	1032	(null)	38	building 1	landmark 1	3379192917	10	Y
2	2590001	1490916	2290703	street_name2	1064	(null)	24	building 2	landmark 2	6837035005	25	Y
3	2590002	1499515	2291438	street_name3	1024	(null)	50	building 3	landmark 3	7518842559	10	Y
4	2590003	1495885	2291947	street_name4	1019	(null)	33	building 4	landmark 4	7397612050	31	Y
5	2590004	1492219	2290720	street_name5	1093	(null)	6	building 5	landmark 5	8965894293	5	Y

Figure 12: Local Address Table

11. **Certification:** This table keeps certification details for each technology such as Certification ID (Primary key), Technology ID (Foreign Key), certificate code, certification name, certification path ID, registration URL, price, currency, exam mode, exam duration, exam duration unit, etc.



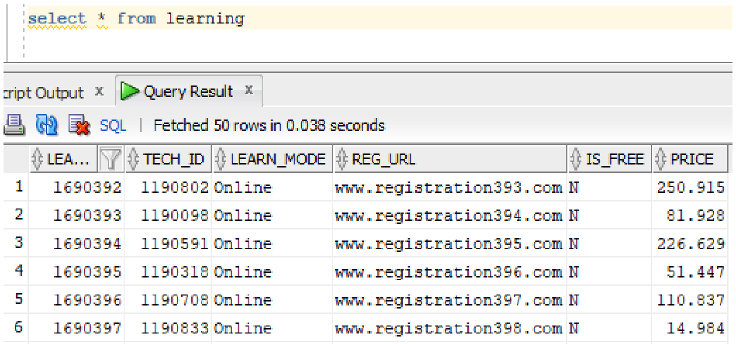
```
select * from certification
```

Fetched 50 rows in 0.04 seconds

	CERT_ID	TECH_ID	CERT_CODE	CERT_NAME	CERT_PATH_ID	CERT_SEQ_IN_PATH	REGISTRATION_URL	PRICE	CURRENCY	EXAM_MODE	EXAM_DURATION	DURATION_UNIT
1	1890218	1190198	(null)	Certification219	(null)	(null)	www.cerification_registration219.com	101	USD	Online	129	Minutes
2	1890219	1190675	(null)	Certification220	(null)	(null)	www.cerification_registration220.com	218	USD	Online	129	Minutes
3	1890220	1190467	(null)	Certification221	(null)	(null)	www.cerification_registration221.com	23	USD	Online	121	Minutes
4	1890221	1190177	(null)	Certification222	(null)	(null)	www.cerification_registration222.com	25	USD	Online	119	Minutes
5	1890222	1190650	(null)	Certification223	(null)	(null)	www.cerification_registration223.com	43	USD	Online	130	Minutes

Figure 13: Certification Table

12. **Learning:** This table keeps data about all the learning paths for each technology such as Learning ID (Primary Key), Technology ID (Foreign Key), Learning mode, registration URL, whether it's free or not (Y/N), price of the learning, etc.



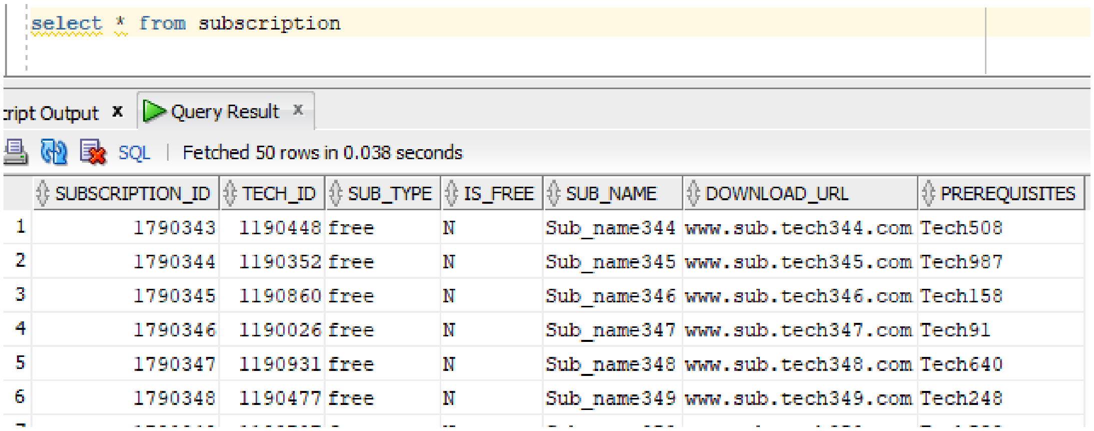
```
select * from learning
```

Fetched 50 rows in 0.038 seconds

	LEA...	TECH_ID	LEARN_MODE	REG_URL	IS_FREE	PRICE
1	1690392	1190802	Online	www.registration393.com	N	250.915
2	1690393	1190098	Online	www.registration394.com	N	81.928
3	1690394	1190591	Online	www.registration395.com	N	226.629
4	1690395	1190318	Online	www.registration396.com	N	51.447
5	1690396	1190708	Online	www.registration397.com	N	110.837
6	1690397	1190833	Online	www.registration398.com	N	14.984

Figure 14: Learning Table

13. **Subscription:** This table keeps data about all the subscriptions customers purchase for technologies such as Subscription ID (Primary Key), Technology ID(Foreign Key), subscription type, whether is free or paid (Y/N), subscription name, URL to download the technology, and any prerequisite technology needed to use this technology.



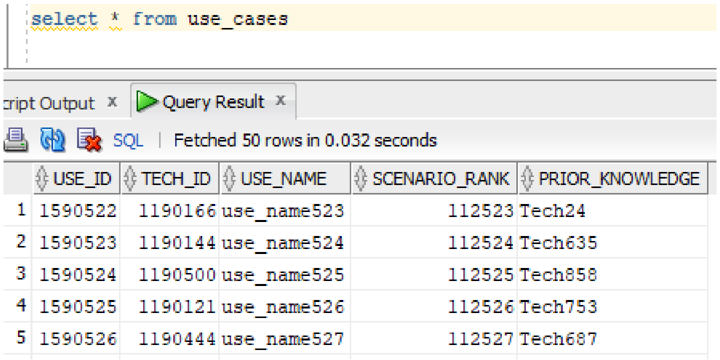
```
select * from subscription
```

Fetched 50 rows in 0.038 seconds

	SUBSCRIPTION_ID	TECH_ID	SUB_TYPE	IS_FREE	SUB_NAME	DOWNLOAD_URL	PREREQUISITES
1	1790343	1190448	free	N	Sub_name344	www.sub.tech344.com	Tech508
2	1790344	1190352	free	N	Sub_name345	www.sub.tech345.com	Tech987
3	1790345	1190860	free	N	Sub_name346	www.sub.tech346.com	Tech158
4	1790346	1190026	free	N	Sub_name347	www.sub.tech347.com	Tech91
5	1790347	1190931	free	N	Sub_name348	www.sub.tech348.com	Tech640
6	1790348	1190477	free	N	Sub_name349	www.sub.tech349.com	Tech248

Figure 15: Subscription Table

14. **Use_cases:** This table keeps data about all the use cases for each technology such as Use case ID (Primary Key), Technology ID (Foreign Key), Use case name, scenario rank in the technology stack, and prior technical knowledge needed for the use case-specific technology, etc.



```
select * from use_cases
```

Fetched 50 rows in 0.032 seconds

	USE_ID	TECH_ID	USE_NAME	SCENARIO_RANK	PRIOR_KNOWLEDGE
1	1590522	1190166	use_name523	112523	Tech24
2	1590523	1190144	use_name524	112524	Tech635
3	1590524	1190500	use_name525	112525	Tech858
4	1590525	1190121	use_name526	112526	Tech753
5	1590526	1190444	use_name527	112527	Tech687

Figure 16: Use cases Table

DATA INTEGRITY

Data Integrity refers to the consistency and maintenance of the data through the life cycle of the database. In a database, data integrity can be ensured through the implementation of Integrity Constraints in a table. Integrity constraints help apply business rules to the database tables. The constraints can either be at a column level or a table level. Some of the most common constraints are,

NOT NULL – Prevents a column from having a NULL value.

PRIMARY KEY – Uniquely identifies each row or record in the table.

FOREIGN KEY – Uniquely identifies a column that references a PRIMARY KEY in another table.

UNIQUE – Prevents a column from having duplicate values.

CHECK – Checks for values that satisfy a specific condition as defined by the user.

Listed below are the constraints that were created for our database development project along with their purpose-

1. **Technology Table**

```
CREATE TABLE db888.technology (
   tech_id        NUMBER(10),
   name           VARCHAR2(50),
   launch_date    DATE,
   is_open_source CHAR(1),
   download_site  VARCHAR2(4000) NOT NULL
);

ALTER TABLE db888.technology ADD CONSTRAINT pk_tech_id PRIMARY
KEY ( tech_id );
```

2. **Technology_Stack Table**

```
CREATE TABLE db888.technology_stack (
   stack_id     NUMBER(10),
   stack_name   VARCHAR2(50) NOT NULL,
   stack_sub_id NUMBER(10) NOT NULL
);

ALTER TABLE db888.technology_stack ADD CONSTRAINT pk_stack_id
PRIMARY KEY ( stack_id );

ALTER TABLE db888..technology_stack ADD CONSTRAINT u_stack_sub_id
UNIQUE ( stack_sub_id );
```

3. Customers Table

```
CREATE TABLE db888.customers (
    cust_id      NUMBER(10),
    initial_name VARCHAR2(10),
    first_name   VARCHAR2(50) NOT NULL,
    middle_name  VARCHAR2(50),
    last_name    VARCHAR2(50) NOT NULL
);
ALTER TABLE db888.customers ADD CONSTRAINT pk_cust_id PRIMARY
KEY ( cust_id );
```

4. Tech_Stack_Map Table

```
CREATE TABLE db888.tech_stack_map (
    tech_id  NUMBER(10),
    stack_id NUMBER(10)
);

ALTER TABLE db888.tech_stack_map
    ADD CONSTRAINT fk_stack_map_tech_id FOREIGN KEY ( tech_id )
        REFERENCES technology ( tech_id );

ALTER TABLE db888.tech_stack_map
    ADD CONSTRAINT fk_stack_map_stack_id FOREIGN KEY ( stack_id )
        REFERENCES technology_stack ( stack_id );
```

5. Use_Cases Table

```
CREATE TABLE db888.use_cases (
    use_id          NUMBER(10),
    tech_id         NUMBER(10) NOT NULL,
    use_name        VARCHAR2(50) NOT NULL,
    scenario_rank   NUMBER(10),
    prior_knowledge VARCHAR2(255) NOT NULL
);

ALTER TABLE db888.use_cases ADD CONSTRAINT pk_use_id PRIMARY KEY
( use_id );

ALTER TABLE db888. use_cases
    ADD CONSTRAINT fk_use_tech_id FOREIGN KEY ( tech_id )
        REFERENCES technology ( tech_id );
```

6. Learning Table

```
CREATE TABLE db888.learning (
   learn_id   NUMBER(10),
   tech_id    NUMBER(10) NOT NULL,
   learn_mode VARCHAR2(10),
   reg_url    VARCHAR2(255) NOT NULL,
   is_free    CHAR(1),
   price      NUMBER(10, 3) NOT NULL
);

ALTER TABLE db888.learning ADD CONSTRAINT pk_learn_id PRIMARY KEY
( learn_id );

ALTER TABLE db888.learning
   ADD CONSTRAINT fk_learn_tech_id FOREIGN KEY ( tech_id )
      REFERENCES technology ( tech_id );
```

7. Subscription Table

```
CREATE TABLE db888.subscription (
   subscription_id NUMBER(10),
   tech_id         NUMBER(10) NOT NULL,
   sub_type        VARCHAR2(20) NOT NULL,
   is_free         CHAR(1),
   sub_name        VARCHAR2(50) NOT NULL,
   download_url    VARCHAR2(255) NOT NULL,
   prerequisites   VARCHAR2(4000) NOT NULL
);

ALTER TABLE db888.subscription ADD CONSTRAINT pk_subscription_id
PRIMARY KEY ( subscription_id );

ALTER TABLE db888. subscription
   ADD CONSTRAINT fk_sub_tech_id FOREIGN KEY ( tech_id )
      REFERENCES technology ( tech_id );
```

8. Certification Table

```
CREATE TABLE db888.certification (
   cert_id          NUMBER(10),
   tech_id          NUMBER(10) NOT NULL,
   cert_code        VARCHAR2(20),
   cert_name        VARCHAR2(50) NOT NULL,
   cert_path_id     NUMBER(2),
   cert_seq_in_path NUMBER(2),
   registration_url VARCHAR2(255) NOT NULL,
   price            NUMBER(10) NOT NULL,
   currency         VARCHAR2(3),
   exam_mode        VARCHAR2(20) NOT NULL,
   exam_duration    NUMBER(3) NOT NULL,
   duration_unit    VARCHAR2(10) NOT NULL
);

ALTER TABLE db888.certification ADD CONSTRAINT pk_certification_id
PRIMARY KEY ( cert_id );

ALTER TABLE db888.certification
   ADD CONSTRAINT fk_cert_tech_id FOREIGN KEY ( tech_id )
      REFERENCES technology ( tech_id );

ALTER TABLE db888.certification ADD CONSTRAINT u_cert_path_id UNIQUE
( cert_path_id );
```

9. Technology_Type Table

```
CREATE TABLE db888.technology_type (
   tech_id NUMBER(10),
   type_id NUMBER(10),
   type    VARCHAR2(20) NOT NULL
);

ALTER TABLE db888.technology_type ADD CONSTRAINT pk_tech_type_id
PRIMARY KEY ( type_id );

ALTER TABLE db888.technology_type
   ADD CONSTRAINT fk_type_tech_id FOREIGN KEY ( tech_id )
      REFERENCES technology ( tech_id );
```

10. Country_Address Table

```
CREATE TABLE db888.country_address (
   country_add_id NUMBER(10),
   country        VARCHAR2(50),
   state          VARCHAR2(50),
   city           VARCHAR2(50),
   area_code      NUMBER(10),
   zipcode        VARCHAR2(10) NOT NULL
);

ALTER TABLE db888.country_address ADD CONSTRAINT pk_country_add_id
PRIMARY KEY ( country_add_id );
```

11. Client_Companies Table

```
CREATE TABLE db888.client_companies (
   comp_id        NUMBER(10),
   country_add_id NUMBER(10),
   comp_name      VARCHAR2(255),
   business_type  VARCHAR2(50) NOT NULL
);

ALTER TABLE db888.client_companies ADD CONSTRAINT pk_cli_comp_id
```

```
PRIMARY KEY ( comp_id );

ALTER TABLE db888.client_companies
   ADD CONSTRAINT fk_cli_comp_add_id FOREIGN KEY ( country_add_id )
      REFERENCES country_address ( country_add_id );
```

12. Owner_Companies Table

```
CREATE TABLE db888.owner_companies (
   o_comp_id      NUMBER(10),
   tech_id        NUMBER(10),
   country_add_id NUMBER(10),
   name           VARCHAR2(50)
);

ALTER TABLE db888.owner_companies ADD CONSTRAINT pk_owner_comp_id
PRIMARY KEY ( o_comp_id );

ALTER TABLE db888.owner_companies
   ADD CONSTRAINT fk_own_comp_tech_id FOREIGN KEY ( tech_id )
      REFERENCES technology ( tech_id );

ALTER TABLE db888.owner_companies
   ADD CONSTRAINT fk_own_comp_add_id FOREIGN KEY ( country_add_id )
      REFERENCES country_address ( country_add_id );
```

13. Tech_Comp_Map Table

```
CREATE TABLE db888.tech_comp_map (
   tech_id NUMBER(10) NOT NULL,
   comp_id NUMBER(10) NOT NULL
);

ALTER TABLE db888.tech_comp_map
   ADD CONSTRAINT fk_comp_map_comp_id FOREIGN KEY ( comp_id )
      REFERENCES client_companies ( comp_id );

ALTER TABLE db888.tech_comp_map
   ADD CONSTRAINT fk_comp_map_tech_id FOREIGN KEY ( tech_id )
      REFERENCES technology ( tech_id );
```

14. Local_Address Table

```
CREATE TABLE db888.local_address (
   local_add_id      NUMBER(10),
   cust_id           NUMBER(10) NOT NULL,
   country_add_id    NUMBER(10) NOT NULL,
   street_name       VARCHAR2(50),
   apartment_num     NUMBER(10),
   office_num        NUMBER(10),
   building_num      NUMBER(10) NOT NULL,
   building_name     VARCHAR2(50),
    landmark           VARCHAR2(255),
    phone              NUMBER(10) NOT NULL,
    isd_code           NUMBER(5),
    is_current_address CHAR(1)
);

ALTER TABLE db888.local_address ADD CONSTRAINT pk_loc_add_id
PRIMARY KEY ( local_add_id );

ALTER TABLE db888.local_address
    ADD CONSTRAINT fk_loc_add_cust_id FOREIGN KEY ( cust_id )
        REFERENCES customers ( cust_id );

ALTER TABLE db888.local_address
    ADD CONSTRAINT fk_loc_add_count_add_id FOREIGN KEY ( country_add_id )
        REFERENCES country_address ( country_add_id );
```

Oracle SQL Developer Data Modeler Summary Report:

```
-- CREATE TABLE------------------------------------------------------------ 14
-- CREATE INDEX------------------------------------------------------------ 5
-- ALTER TABLE------------------------------------------------------------- 28
-- CREATE VIEW------------------------------------------------------------- 4
-- ALTER VIEW--------------------------------------------------------------  0
-- CREATE PACKAGE---------------------------------------------------------- 0
-- CREATE PACKAGE BODY----------------------------------------------------- 0
-- CREATE PROCEDURE-------------------------------------------------------  5
-- CREATE FUNCTION-------------------------------------------------------   5
-- CREATE TRIGGER---------------------------------------------------------- 0
-- ALTER TRIGGER----------------------------------------------------------- 0
-- CREATE COLLECTION TYPE-------------------------------------------------- 0
-- CREATE STRUCTURED TYPE-------------------------------------------------- 0
-- CREATE STRUCTURED TYPE BODY--------------------------------------------  0
-- CREATE CLUSTER---------------------------------------------------------  0
-- CREATE CONTEXT---------------------------------------------------------- 0
-- CREATE DATABASE--------------------------------------------------------- 0
-- CREATE DIMENSION-------------------------------------------------------- 0
-- CREATE DIRECTORY-------------------------------------------------------  0
-- CREATE DISK GROUP------------------------------------------------------  0
-- CREATE ROLE------------------------------------------------------------  0
-- CREATE ROLLBACK SEGMENT------------------------------------------------- 0
-- CREATE SEQUENCE-------------------------------------------------------- 14
-- CREATE MATERIALIZED VIEW------------------------------------------------ 0
-- CREATE MATERIALIZED VIEW LOG-------------------------------------------- 0
-- CREATE SYNONYM---------------------------------------------------------- 0
-- CREATE TABLESPACE------------------------------------------------------- 0
-- CREATE USER------------------------------------------------------------  0
-- DROP TABLESPACE--------------------------------------------------------  0
-- DROP DATABASE----------------------------------------------------------  0
-- REDACTION POLICY-------------------------------------------------------  0
-- TSDP POLICY------------------------------------------------------------  0
-- ORDS DROP SCHEMA---------------------------------------------------------  0
-- ORDS ENABLE SCHEMA-----------------------------------------------------  0
-- ORDS ENABLE OBJECT------------------------------------------------------- 0
-- ERRORS----------------------------------------------------------------------------- 0
-- WARNINGS------------------------------------------------------------------------ 1
```

DATA SYNTHESIS

The data for the project has been synthesized using PL/SQL blocks. The scripts are mentioned below. DBMS_RANDOM package is used to prepare random samples of data.

We also have used sequences for generating synthetic keys for all tables.

SEQUENCES

1. create sequence **SEQ_TECHNOLOGY_TECH_ID** start with 1190000 increment by 1 nocycle;
2. create sequence **SEQ_TECH_STACK_ID** start with 1290000 increment by 1 nocycle;
3. create sequence **SEQ_TECH_STACK_SUB_ID** start with 1390000 increment by 1 nocycle;
4. create sequence **SEQ_CUST_ID** start with 1490000 increment by 1 nocycle;
5. create sequence **SEQ_USE_ID** start with 1590000 increment by 1 nocycle;
6. create sequence **SEQ_LEARN_ID** start with 1690000 increment by 1 nocycle;
7. create sequence **SEQ_SUBSCRIPTION_ID** start with 1790000 increment by 1 nocycle;
8. create sequence **SEQ_CERT_ID** start with 1890000 increment by 1 nocycle;
9. create sequence **SEQ_SCERT_PATH_ID** start with 1990000 increment by 1 nocycle;
10. create sequence **SEQ_TYPE_ID** start with 2190000 increment by 1 nocycle;
11. create sequence **SEQ_COUNTRY_ADD_ID** start with 2290000 increment by 1 nocycle;
12. create sequence **SEQ_OWN_COMP_ID** start with 2390000 increment by 1 nocycle;
13. create sequence **SEQ_LOC_COMP_ID** start with 2490000 increment by 1 nocycle;
14. create sequence **SEQ_LOC_ADD_ID** start with 2590000 increment by 1 nocycle;

DATA INSERTION SCRIPTS

1. Technology Table

```
DECLARE
  v_name          VARCHAR2(20);
  v_is_open_source CHAR(1) := 'Y';
  v_download_site  VARCHAR2(100);
  v_sql           VARCHAR2(2000);
BEGIN
  FOR i IN 1..1000 LOOP
    v_name := 'Tech' || i;
    v_download_site := 'www.tech'
                     || i
```

```
                    || '.com';
      v_sql := q'{insert into
technology(tech_id,name,launch_date,is_open_source,download_site)
          values(:1,:2,:3,:4,:5)}';
      EXECUTE IMMEDIATE v_sql
         USING seq_technology_tech_id.nextval, v_name, sysdate, v_is_open_source,
v_download_site;
      COMMIT;
   END LOOP;
END;
```

2. Tech_Stack Table

```
DECLARE
   v_stack_name VARCHAR2(20);
   v_sql        VARCHAR2(2000);
BEGIN
   FOR i IN 1..500 LOOP
      v_stack_name := 'Tech_Stack' || i;
      v_sql := q'{insert into technology_stack(stack_id,stack_name,stack_sub_id)
          values(:1,:2,:3)}';
      EXECUTE IMMEDIATE v_sql
         USING seq_tech_stack_id.nextval, v_stack_name,
seq_tech_stack_sub_id.nextval;
      COMMIT;
   END LOOP;
END;
```

3. Subscription Table

```
DECLARE
   v_sub_type      VARCHAR2(20) := 'free';
   v_is_free       CHAR(1) := 'N';
   v_sub_name      VARCHAR2(50);
   v_download_url  VARCHAR2(255);
   v_prerequisites VARCHAR2(4000);
   v_sql           VARCHAR2(2000);
   v_tech_id       NUMBER(10);
BEGIN
   FOR i IN 1..10000 LOOP
      v_sub_name := 'Sub_name' || i;
      v_download_url := 'www.sub.tech'
```

```
             || i
             || '.com';
    v_prerequisites := 'Tech'
              || round(dbms_random.value(1, 1000));
    SELECT
      tech_id
    INTO v_tech_id
    FROM
      (
        SELECT
          tech_id
        FROM
          technology
        ORDER BY
          dbms_random.value
      )
    WHERE
      ROWNUM = 1;

    v_sql := q'{insert into subscription(subscription_id, tech_id, sub_type, is_free, sub_name, download_url, prerequisites)
       values(:1,:2,:3,:4,:5,:6,:7)}';
    EXECUTE IMMEDIATE v_sql
      USING seq_subscription_id.nextval, v_tech_id, v_sub_type, v_is_free, v_sub_name, v_download_url, v_prerequisites;

    COMMIT;
  END LOOP;
END;
```

4. **Use_Cases Table**

```
DECLARE
  v_use_name    VARCHAR2(50);
  v_prior_knowledge VARCHAR2(255);
  v_scenario_rank  NUMBER(10);
  v_tech_id     NUMBER(10);
  v_sql        VARCHAR2(2000);
BEGIN
  FOR i IN 1..4000 LOOP
    v_use_name := 'use_name' || i;
```

```
      v_scenario_rank := '112' || i;
      v_prior_knowledge := 'Tech'
                     || round(dbms_random.value(1, 1000));
      SELECT
         tech_id
      INTO v_tech_id
      FROM
         (
            SELECT
               tech_id
            FROM
               technology
            ORDER BY
               dbms_random.value
         )
      WHERE
         ROWNUM = 1;

      v_sql := q'{insert into use_cases(use_id, tech_id, use_name, scenario_rank,
prior_knowledge)
          values(:1,:2,:3,:4,:5)}';
      EXECUTE IMMEDIATE v_sql
         USING seq_use_id.nextval, v_tech_id, v_use_name, v_scenario_rank,
v_prior_knowledge;
      COMMIT;
   END LOOP;
END;
```

5. Learning Table

```
DECLARE
   v_learn_mode VARCHAR2(10) := 'Online';
   v_reg_url    VARCHAR2(255);
   v_is_free    CHAR(1);
   v_price      NUMBER(10, 3);
   v_tech_id    NUMBER(10);
   v_sql        VARCHAR2(2000);
BEGIN
   FOR i IN 1..5000 LOOP
      v_reg_url := 'www.registration'
                || i
```

```
            || '.com';
    v_price := dbms_random.value(0, 300);
    IF v_price = 0 THEN
      v_is_free := 'Y';
    ELSE
      v_is_free := 'N';
    END IF;
SELECT
      tech_id
    INTO v_tech_id
    FROM
      (
        SELECT
          tech_id
        FROM
          technology
        ORDER BY
          dbms_random.value
      )
    WHERE
      ROWNUM = 1;

    v_sql := q'{insert into learning(learn_id, tech_id, learn_mode, reg_url, is_free,price)
       values(:1,:2,:3,:4,:5,:6)}';
    EXECUTE IMMEDIATE v_sql
      USING seq_learn_id.nextval, v_tech_id, v_learn_mode, v_reg_url, v_is_free, v_price;
    COMMIT;
  END LOOP;
END;
```

6. Tech_Stack_Map Table

```
DECLARE
  v_stack_id NUMBER(10);
  v_tech_id  NUMBER(10);
  v_sql      VARCHAR2(2000);
BEGIN
  FOR i IN 1..2000 LOOP
    SELECT
      tech_id
      INTO v_tech_id
      FROM
         (
            SELECT
               tech_id
            FROM
               technology
            ORDER BY
               dbms_random.value
         )
      WHERE
         ROWNUM = 1;

      SELECT
         stack_id
      INTO v_stack_id
      FROM
         (
            SELECT
               stack_id
            FROM
               technology_stack
            ORDER BY
               dbms_random.value
         )
      WHERE
         ROWNUM = 1;

      v_sql := q'{insert into tech_stack_map(tech_id, stack_id)
          values(:1,:2)}';
      EXECUTE IMMEDIATE v_sql
         USING v_tech_id, v_stack_id;
      COMMIT;
   END LOOP;
END;
```

7. Customers Table

```
DECLARE
   v_initial    CHAR(5) := 'Mr.';
   v_first_name VARCHAR2(50);
   v_last_name  VARCHAR2(50);
   v_sql        VARCHAR2(2000);
BEGIN
   FOR i IN 1..15000 LOOP
      v_first_name := 'first name' || i;
      v_last_name := 'last name' || i;
      v_sql := q'{insert into customers(cust_id,initial_name,first_name,last_name)
           values(:1,:2,:3,:4)}';
      EXECUTE IMMEDIATE v_sql
         USING seq_cust_id.nextval, v_initial, v_first_name, v_last_name;
      COMMIT;
   END LOOP;
END;
```

8. Certification Table

```
DECLARE
   v_tech_id          NUMBER(10);
   v_cert_name        VARCHAR2(50);
   v_registration_url VARCHAR2(255);
   v_price            NUMBER(10, 3);
   v_currency         VARCHAR2(5) := 'USD';
   v_exam_mode        VARCHAR2(10) := 'Online';
   v_exam_duration    NUMBER(3);
   v_duration_unit    VARCHAR2(10) := 'Minutes';
   v_sql              VARCHAR2(2000);
BEGIN
   FOR i IN 1..3000 LOOP
      SELECT
         tech_id
      INTO v_tech_id
      FROM
         (
            SELECT
               tech_id
            FROM
               technology
            ORDER BY
               dbms_random.value
         )
      WHERE
         ROWNUM = 1;

      v_cert_name := 'Certification' || i;
      v_registration_url := 'www.cerification_registration'
                      || i
                      || '.com';
      v_price := dbms_random.value(0, 300);
      v_exam_duration := round(dbms_random.value(60, 180));
      v_sql := q'{insert into certification(cert_id, tech_id, cert_name, registration_url,
price, currency, exam_mode, exam_duration, duration_unit)
           values(:1,:2,:3,:4,:5,:6,:7,:8,:9)}';
      EXECUTE IMMEDIATE v_sql
         USING seq_cert_id.nextval, v_tech_id, v_cert_name, v_registration_url, v_price,
v_currency, v_exam_mode, v_exam_duration,
         v_duration_unit;

      COMMIT;
   END LOOP;
END;
```

9. Technology_Type Table

```
DECLARE
   v_tech_id VARCHAR2(20);
   v_type    VARCHAR2(20);
   v_sql     VARCHAR2(2000);
BEGIN
   FOR i IN 1..4000 LOOP
      SELECT
         tech_id
      INTO v_tech_id
      FROM
         (
            SELECT
               tech_id
            FROM
               technology
```

```
                ORDER BY
                    dbms_random.value
            )
        WHERE
            ROWNUM = 1;

        v_type := 'Type_' || i;
        v_sql := q'{insert into technology_type(type_id,tech_id,type)
              values(:1,:2,:3)}';
        EXECUTE IMMEDIATE v_sql
            USING seq_type_id.nextval, v_tech_id, v_type;
        COMMIT;
    END LOOP;
END;
```

10. Country_Address Table

```
DECLARE
    v_country VARCHAR2(50);
    v_state   VARCHAR2(50);
    v_city    VARCHAR2(50);
    v_zipcode NUMBER(10);
    v_sql     VARCHAR2(2000);
BEGIN
    FOR i IN 1..2000 LOOP
        v_country := 'Country' || i;
        v_state := 'State' || i;
        v_city := 'City' || i;
        v_zipcode := dbms_random.value(30000, 50000);
        v_sql := q'{insert into country_address(country_add_id, country, state, city, zipcode)
              values(:1,:2,:3,:4,:5)}';
        EXECUTE IMMEDIATE v_sql
            USING seq_country_add_id.nextval, v_country, v_state, v_city, v_zipcode;
        COMMIT;
    END LOOP;
END;
```

11. Client_Companies Table

```
DECLARE
   v_comp_id        NUMBER(10);
   v_country_add_id NUMBER(10);
   v_comp_name      VARCHAR2(255);
   v_business_type  VARCHAR2(50);
   v_sql            VARCHAR2(2000);
BEGIN
   FOR i IN 1..2000 LOOP
      v_comp_name := 'client_cmp_' || i;
      v_business_type := 'Business' || i;
      SELECT
         country_add_id
      INTO v_country_add_id
      FROM
         (
            SELECT
               country_add_id
            FROM
               country_address
            ORDER BY
               dbms_random.value
         )
      WHERE
         ROWNUM = 1;

      v_sql := q'{insert into client_companies(comp_id, country_add_id, comp_name,
business_type)
          values(:1,:2,:3,:4)}';
      EXECUTE IMMEDIATE v_sql
         USING seq_country_add_id.nextval, v_country_add_id, v_comp_name,
v_business_type;
      COMMIT;
   END LOOP;
END;
```

12. Owner_Companies Table

```
DECLARE
   v_o_comp_id      NUMBER(10);
   v_tech_id        NUMBER(10);
   v_country_add_id NUMBER(10);
   v_name           VARCHAR2(50);
   v_sql            VARCHAR2(2000);
BEGIN
   FOR i IN 1..200 LOOP
      v_name := 'Owner_comp_' || i;
      SELECT
         country_add_id
      INTO v_country_add_id
      FROM
         (
            SELECT
               country_add_id
            FROM
               country_address
            ORDER BY
               dbms_random.value
         )
      WHERE
         ROWNUM = 1;

      SELECT
         tech_id
      INTO v_tech_id
      FROM
         (
            SELECT
               tech_id
            FROM
               technology
            ORDER BY
               dbms_random.value
         )
      WHERE
         ROWNUM = 1;
```

```
        v_sql := q'{insert into owner_companies(o_comp_id,tech_id, country_add_id, name)
           values(:1,:2,:3,:4)}';
        EXECUTE IMMEDIATE v_sql
          USING seq_own_comp_id.nextval, v_tech_id, v_country_add_id, v_name;
        COMMIT;
    END LOOP;
END;
```

13. Local_Address Table

```
DECLARE
  v_local_add_id      NUMBER(10);
  v_cust_id           NUMBER(10);
  v_country_add_id    NUMBER(10);
  v_street_name       VARCHAR2(50);
  v_apartment_num     NUMBER(10);
  v_building_num      NUMBER(10);
  v_building_name     VARCHAR2(50);
  v_landmark          VARCHAR2(255);
  v_phone             NUMBER(10);
  v_isdcode           NUMBER(5);
  v_is_currentaddress CHAR(1) := 'Y';
  v_sql               VARCHAR2(2000);
BEGIN
  FOR i IN 1..5000 LOOP
    v_street_name := 'street_name' || i;
    v_apartment_num := dbms_random.value(1001, 1099);
    v_building_num := dbms_random.value(1, 50);
    v_building_name := 'building ' || i;
    v_landmark := 'landmark ' || i;
    v_phone := dbms_random.value(3111111111, 9876543210);
    v_isdcode := dbms_random.value(1, 100);
    SELECT
      country_add_id
    INTO v_country_add_id
    FROM
      (
        SELECT
          country_add_id
        FROM
```

```
          country_address
        ORDER BY
          dbms_random.value
      )
    WHERE
      ROWNUM = 1;

    SELECT
      cust_id
    INTO v_cust_id
    FROM
      (
        SELECT
          cust_id
        FROM
          customers
        ORDER BY
          dbms_random.value
      )
    WHERE
      ROWNUM = 1;

    v_sql := q'{insert into local_address(
    local_add_id,
    cust_id,
    country_add_id,
    street_name,
    apartment_num,
    building_num,
    building_name,
    landmark,
    phone,
    ISD_code,
    is_current_address)
       values(:1,:2,:3,:4,:5,:6,:7,:8,:9,:10,:11)}';
    EXECUTE IMMEDIATE v_sql
      USING seq_loc_add_id.nextval, v_cust_id, v_country_add_id, v_street_name,
v_apartment_num, v_building_num, v_building_name,
      v_landmark, v_phone, v_isdcode, v_is_currentaddress;
      COMMIT;
   END LOOP;
END;
```

14. Tech_Stack_Map Table

```
DECLARE
   v_comp_id NUMBER(10);
   v_tech_id  NUMBER(10);
   v_sql      VARCHAR2(2000);
BEGIN
   FOR i IN 1..4000 LOOP
      SELECT
         tech_id
      INTO v_tech_id
      FROM
         (
            SELECT
               tech_id
            FROM
               technology
            ORDER BY
               dbms_random.value
         )
      WHERE
         ROWNUM = 1;

      SELECT
         comp_id
      INTO v_comp_id
      FROM
         (
            SELECT
               comp_id
            FROM
               client_companies
            ORDER BY
               dbms_random.value
         )
      WHERE
         ROWNUM = 1;
```

```
          v_sql := q'{insert into tech_comp_map(tech_id, comp_id)
                values(:1,:2)}';
          EXECUTE IMMEDIATE v_sql
              USING v_tech_id, v_comp_id;
          COMMIT;
      END LOOP;
  END;
```

The tabulation below provides a summary of the data housed in the tables,

Table 1: Summary of Data housed in a table

Table Name	**Columns**	**Number of constraints**	**Name of sequence**	**Number of Records**
TECHNOLOGY	5	1	SEQ_TECHNOLOGY_TECH_	1000
TECHNOLOGY_TYPE	3	2	SEQ_TYPE_ID	400
TECHNOLOGY_STACK	3	2	SEQ_TECH_STACK_ID SEQ_TECH_STACK_SUB_ID	2000
TECH_STACK_MAP	2	2	-	2000
OWNER_COMPANIES	4	3	SEQ_OWN_COMP_ID	200
CLIENT_COMPANIES	4	2	SEQ_LOC_COMP_ID	2000
TECH_COMP_MAP	2	2	-	4000
COUNTRY_ADDRESS	6	1	SEQ_COUNTRY_ADD_ID	1000
LOCAL_ADDRESS	12	3	SEQ_LOC_ADD_ID	5000
CUSTOMERS	5	1	SEQ_CUST_ID	10000
CERTIFICATION	12	3	SEQ_CERT_ID SEQ_SCERT_PATH_ID	3000
LEARNING	6	2	SEQ_LEARN_ID	5000
SUBSCRIPTION	7	2	SEQ_SUBSCRIPTION_ID	10000
USE_CASES	5	2	SEQ_USE_ID	4000

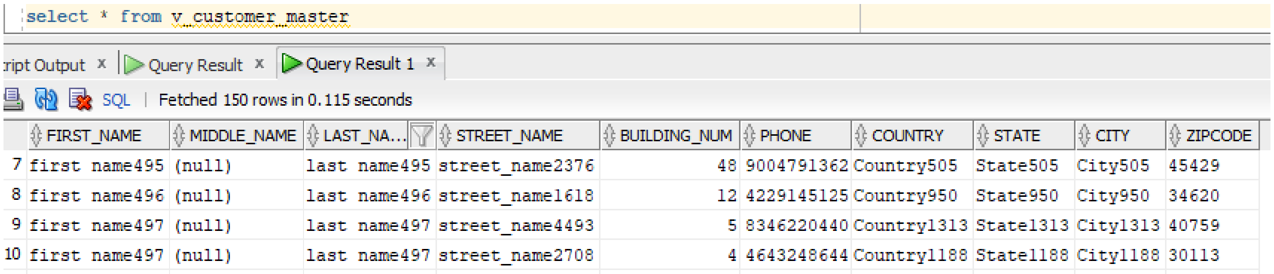
VIEW/FUNCTION/PROCEDURE WRITING

We have written a few views, functions, and procedures which can be used for various queries or tasks. The queries and tasks can be executed from any web portal which will be designed based on this database.

VIEWS

1. **Fetch details about any customer such as name, address, and country.**

```
CREATE OR REPLACE VIEW v_customer_master AS
    SELECT
        c.first_name,
        c.middle_name,
        c.last_name,
        la.street_name,
        la.building_num,
        la.phone,
        ca.country,
        ca.state,
        ca.city,
        ca.zipcode
    FROM
        customers       c
        LEFT JOIN local_address   la ON ( c.cust_id = la.cust_id )
        LEFT JOIN country_address ca ON ( la.country_add_id = ca.country_add_id );
```



```
select * from v_customer_master
```

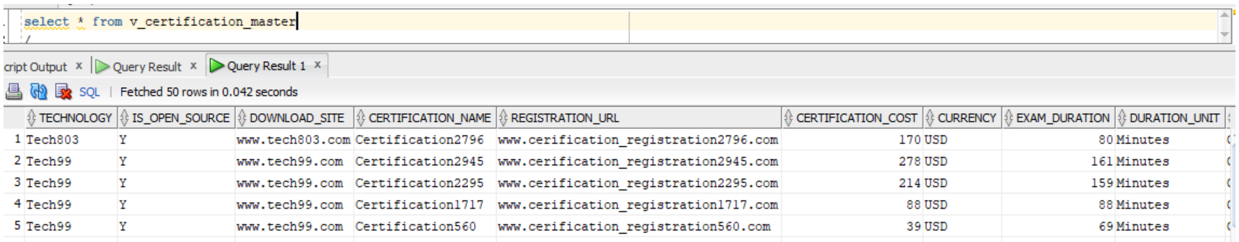
Fetched 150 rows in 0.115 seconds

	FIRST_NAME	MIDDLE_NAME	LAST_NA...	STREET_NAME	BUILDING_NUM	PHONE	COUNTRY	STATE	CITY	ZIPCODE
7	first name495	(null)	last name495	street_name2376	48	9004791362	Country505	State505	City505	45429
8	first name496	(null)	last name496	street_name1618	12	4229145125	Country950	State950	City950	34620
9	first name497	(null)	last name497	street_name4493	5	8346220440	Country1313	State1313	City1313	40759
10	first name497	(null)	last name497	street_name2708	4	4643248644	Country1188	State1188	City1188	30113

Figure 17: Customer_master view

2. **Fetch certification details of any technology such as technology name, price, certification name, learning sites, etc.**

```
CREATE OR REPLACE VIEW v_certification_master AS
    SELECT
        t.name       AS technology,
        t.is_open_source,
        t.download_site,
        c.cert_name  AS certification_name,
        c.registration_url,
        c.price      AS certification_cost,
        c.currency,
        c.exam_duration,
        c.duration_unit,
        l.learn_mode AS learning_mode,
        l.reg_url     AS learning_site,
        l.price       AS learning_cost
    FROM
        technology    t
        LEFT JOIN certification c ON ( t.tech_id = c.tech_id )
        LEFT JOIN learning       l ON ( t.tech_id = l.tech_id );
```



```
select * from v_certification_master
```

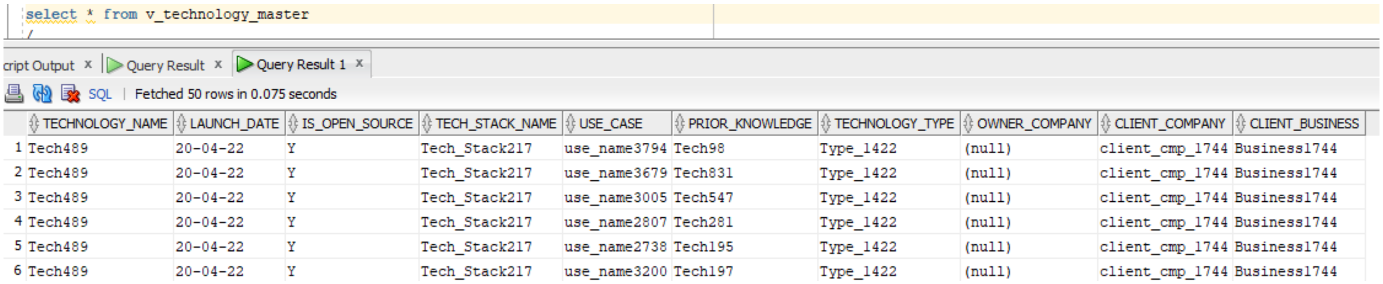
Fetched 50 rows in 0.042 seconds

	TECHNOLOGY	IS_OPEN_SOURCE	DOWNLOAD_SITE	CERTIFICATION_NAME	REGISTRATION_URL	CERTIFICATION_COST	CURRENCY	EXAM_DURATION	DURATION_UNIT
1	Tech803	Y	www.tech803.com	Certification2796	www.cerification_registration2796.com	170	USD	80	Minutes
2	Tech99	Y	www.tech99.com	Certification2945	www.cerification_registration2945.com	278	USD	161	Minutes
3	Tech99	Y	www.tech99.com	Certification2295	www.cerification_registration2295.com	214	USD	159	Minutes
4	Tech99	Y	www.tech99.com	Certification1717	www.cerification_registration1717.com	88	USD	88	Minutes
5	Tech99	Y	www.tech99.com	Certification560	www.cerification_registration560.com	39	USD	69	Minutes

Figure 18: Certification_master view

3. **Fetch technology details such as technology name, technology stack name, owner company name and client company name, etc.**

```
CREATE OR REPLACE VIEW v_technology_master AS
    SELECT
        a.name          AS technology_name,
        a.launch_date,
        a.is_open_source,
        b.stack_name    AS tech_stack_name,
        d.use_name      AS use_case,
        d.prior_knowledge,
        e.type          AS technology_type,
        f.name          AS owner_company,
        g.comp_name     AS client_company,
        g.business_type AS client_business
    FROM
        technology       a
        LEFT JOIN tech_stack_map    c ON ( c.tech_id = a.tech_id )
        LEFT JOIN technology_stack b ON ( b.stack_id = c.stack_id )
        LEFT JOIN use_cases         d ON ( a.tech_id = d.tech_id )
        LEFT JOIN technology_type  e ON ( a.tech_id = e.tech_id )
        LEFT JOIN owner_companies  f ON ( a.tech_id = f.tech_id )
        LEFT JOIN tech_comp_map     h ON ( a.tech_id = h.tech_id )
        LEFT JOIN client_companies g ON ( g.comp_id = h.comp_id );
```



```
select * from v_technology_master
/
```

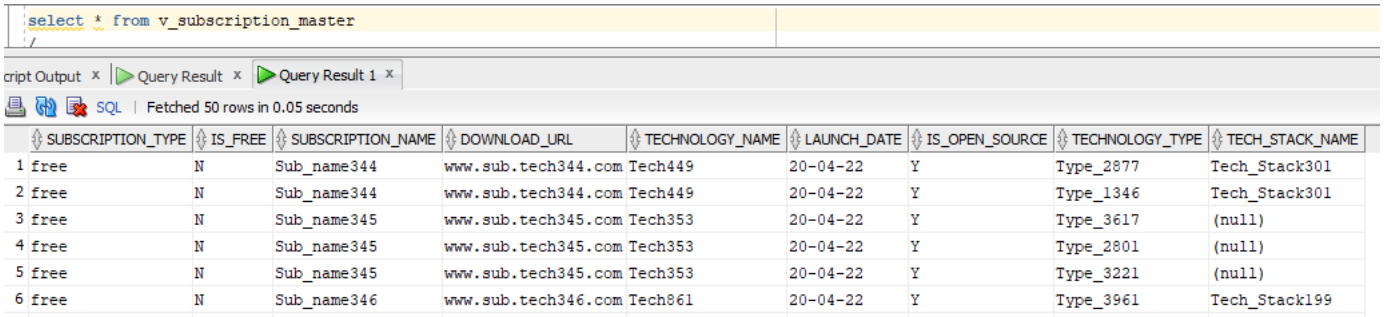
Fetched 50 rows in 0.075 seconds

	TECHNOLOGY_NAME	LAUNCH_DATE	IS_OPEN_SOURCE	TECH_STACK_NAME	USE_CASE	PRIOR_KNOWLEDGE	TECHNOLOGY_TYPE	OWNER_COMPANY	CLIENT_COMPANY	CLIENT_BUSINESS
1	Tech489	20-04-22	Y	Tech_Stack217	use_name3794	Tech98	Type_1422	(null)	client_cmp_1744	Business1744
2	Tech489	20-04-22	Y	Tech_Stack217	use_name3679	Tech831	Type_1422	(null)	client_cmp_1744	Business1744
3	Tech489	20-04-22	Y	Tech_Stack217	use_name3005	Tech547	Type_1422	(null)	client_cmp_1744	Business1744
4	Tech489	20-04-22	Y	Tech_Stack217	use_name2807	Tech281	Type_1422	(null)	client_cmp_1744	Business1744
5	Tech489	20-04-22	Y	Tech_Stack217	use_name2738	Tech195	Type_1422	(null)	client_cmp_1744	Business1744
6	Tech489	20-04-22	Y	Tech_Stack217	use_name3200	Tech197	Type_1422	(null)	client_cmp_1744	Business1744

Figure 19: Technology_master view

4. **Fetch subscription details such as technology name, technology stack name and subscription name, etc.**

```
CREATE OR REPLACE VIEW v_subscription_master AS
    SELECT
        s.sub_type      AS subscription_type,
        s.is_free,
        s.sub_name      AS subscription_name,
        s.download_url,
        t.name          AS technology_name,
        t.launch_date,
        t.is_open_source,
        tt.type         AS technology_type,
        ts.stack_name AS tech_stack_name
    FROM
        subscription      s
        LEFT JOIN technology        t ON ( t.tech_id = s.tech_id )
        LEFT JOIN technology_type  tt ON ( tt.tech_id = s.tech_id )
        LEFT JOIN tech_stack_map   tsm ON ( tsm.tech_id = s.tech_id )
        LEFT JOIN technology_stack ts ON ( ts.stack_id = tsm.stack_id );
```



```
select * from v_subscription_master
/
```

Fetched 50 rows in 0.05 seconds

	SUBSCRIPTION_TYPE	IS_FREE	SUBSCRIPTION_NAME	DOWNLOAD_URL	TECHNOLOGY_NAME	LAUNCH_DATE	IS_OPEN_SOURCE	TECHNOLOGY_TYPE	TECH_STACK_NAME
1	free	N	Sub_name344	www.sub.tech344.com	Tech449	20-04-22	Y	Type_2877	Tech_Stack301
2	free	N	Sub_name344	www.sub.tech344.com	Tech449	20-04-22	Y	Type_1346	Tech_Stack301
3	free	N	Sub_name345	www.sub.tech345.com	Tech353	20-04-22	Y	Type_3617	(null)
4	free	N	Sub_name345	www.sub.tech345.com	Tech353	20-04-22	Y	Type_2801	(null)
5	free	N	Sub_name345	www.sub.tech345.com	Tech353	20-04-22	Y	Type_3221	(null)
6	free	N	Sub_name346	www.sub.tech346.com	Tech861	20-04-22	Y	Type_3961	Tech_Stack199

Figure 20: Subscription_master view

FUNCTIONS

1. **Get the price of certification when certification ID and ‘S’ are passed as arguments where S stands for single. When technology ID and ‘F’ are passed as arguments where F stands for full price, the total price of all available certifications of the technology is returned.**

```
CREATE OR REPLACE FUNCTION f_certification_price (
   v_id IN NUMBER,
   ind  IN CHAR
) RETURN NUMBER IS

   v_tech_id NUMBER(10);
   v_cert_id NUMBER(10);
   v_ind     CHAR(1) := ind;
   v_price   NUMBER(10, 3);
BEGIN
   IF v_ind = 'S' THEN
      v_cert_id := v_id;
   ELSIF v_ind = 'F' THEN
      v_tech_id := v_id;
   ELSE
      NULL;
   END IF;

   IF
      v_tech_id IS NULL
      AND v_cert_id IS NOT NULL
      AND v_ind = 'S'
   THEN
      SELECT
         price
      INTO v_price
      FROM
         certification
      WHERE
         cert_id = v_cert_id;

   ELSIF
      v_tech_id IS NOT NULL
      AND v_cert_id IS NULL
      AND v_ind = 'F'
   THEN
      SELECT
         SUM(c.price)
      INTO v_price
      FROM
         certification c,
         technology    t
      WHERE
            t.tech_id = v_tech_id
         AND c.tech_id = t.tech_id;
```

```
    END IF;

    IF v_price IS NOT NULL THEN
        RETURN v_price;
    ELSE
        RETURN -9;
    END IF;
END;
```

2. **Get the price of learning when learning ID and ‘S’ are passed as arguments where S stands for single. When technology ID and ‘F’ are passed as arguments where F stands for full price, the total price of all available certifications of the technology is returned.**

```
CREATE OR REPLACE FUNCTION f_learning_price (
    id IN NUMBER,
    ind  IN CHAR
) RETURN NUMBER IS

    v_tech_id NUMBER(10);
    v_learn_id NUMBER(10);
    v_ind     CHAR(1) := ind;
    v_price   NUMBER(10, 3);
BEGIN
    IF v_ind = 'S' THEN
        v_learn_id := id;
    ELSIF v_ind = 'F' THEN
        v_tech_id := id;
    END IF;

    IF
            v_learn_id IS NOT NULL
        AND v_ind = 'S'
    THEN
        SELECT
            price
        INTO v_price
        FROM
            learning
        WHERE
            learn_id = v_learn_id;

    ELSIF
        v_tech_id IS NOT NULL
        AND v_ind = 'F'
    THEN
        SELECT
            SUM(l.price)
        INTO v_price
        FROM
            learning l,
            technology  t
        WHERE
                t.tech_id = v_tech_id
            AND l.tech_id = l.tech_id;

    END IF;

    IF v_price IS NOT NULL THEN
        RETURN v_price;
    ELSE
        RETURN -9;
    END IF;
END;
```

3. **Get the company list of a given technology ID. When the argument is "O" get the owner company name and when the argument is "C" get the list of client company names separated by a comma.**

```
CREATE OR REPLACE FUNCTION f_company_list (
    v_id IN NUMBER,
    ind  IN CHAR
) RETURN VARCHAR2 IS

    comp_id        NUMBER(10);
    country_add_id NUMBER(10);
    comp_name      VARCHAR2(255);
    business_type  VARCHAR2(50);
    o_comp_id      NUMBER(10);
    tech_id        NUMBER(10);
    name           VARCHAR2(50);
    v_ind          CHAR(1) := ind;
    v_name         VARCHAR2(4000);
BEGIN
    IF v_ind = 'O' THEN
        SELECT
            comp_name
        INTO v_name
        FROM
            owner_companies
        WHERE
            tech_id = v_id;

    ELSIF v_ind = 'C' THEN
        SELECT
            rtrim(
                LISTAGG(c.comp_name, ',') WITHIN GROUP(
                ORDER BY
                    t.tech_id
                )
            ) "comp_name"
        INTO v_name
        FROM
```

```
         client_companies c,
         tech_comp_map    t
      WHERE
            c.comp_id = t.comp_id
         AND t.tech_id = v_id;

   END IF;

   IF v_name IS NOT NULL THEN
      RETURN v_name;
   ELSE
      RETURN -9;
   END IF;
END;
```

4. **Get the full name of the customer in the camel case. The first name or last name is given as an argument.**

```
CREATE OR REPLACE FUNCTION f_search_customer (
   v_name IN VARCHAR2
) RETURN VARCHAR2 IS
   v_full_name VARCHAR2(100);
BEGIN
   IF v_name IS NOT NULL THEN
      SELECT
         first_name
         || ' '
         || last_name
      INTO v_full_name
      FROM
         customers c
      WHERE
         c.first_name = v_name
         OR c.last_name = v_name;

      RETURN initcap(v_full_name);
   ELSE
      RETURN -9;
   END IF;
END;
```

5. **Get the address of the customer including country and city details.**

```
CREATE OR REPLACE FUNCTION f_customer_address (
   v_customer_id IN NUMBER
) RETURN VARCHAR2 IS
   v_address VARCHAR2(4000);
BEGIN
   IF v_customer_id IS NOT NULL THEN
      SELECT
         la.street_name
         || ' '
         || la.apartment_num
         || ' '
         || la.office_num
         || ' '
         || la.building_num
         || ' '
         || la.building_name
         || ' '
         || ca.country
         || ' '
         || ca.state
         || ' '
         || ca.city
         || ' '
         || ca.zipcode
      INTO v_address
      FROM
           country_address ca
         JOIN local_address la ON ( ca.country_add_id = la.country_add_id )
                      AND la.cust_id = v_customer_id;

      IF replace(v_address, ' ', '') = '' THEN
         RETURN 'Not Found';
      ELSE
         RETURN v_address;
      END IF;

   ELSE
      RETURN 'Not Found';
   END IF;
END;
```

PROCEDURES

1. **Insert new learning data.**

```
CREATE OR REPLACE PROCEDURE p_insert_learning (
  p_tech_id   IN NUMBER,
  p_learn_mode IN VARCHAR2,
  p_reg_url   IN VARCHAR2,
  p_price     NUMBER,
  p_message   OUT VARCHAR2
) IS
  v_is_free CHAR(1);
  v_cnt    NUMBER(2);
BEGIN
  SELECT
```

```
        COUNT(1)
    INTO v_cnt
    FROM
        technology
    WHERE
        tech_id = p_tech_id;

    IF v_cnt > 0 THEN
        IF p_price IS NULL OR p_price = 0 THEN
            v_is_free := 'Y';
        ELSE
            v_is_free := 'N';
        END IF;

        INSERT INTO learning (
            learn_id,
            tech_id,
            learn_mode,
            reg_url,
            is_free,
            price
        ) VALUES (
            seq_learn_id.NEXTVAL,
            p_tech_id,
            p_learn_mode,
            p_reg_url,
            v_is_free,
            p_price
        );

        COMMIT;
        p_message := 'Added Successfully!!!';
    ELSE
        p_message := 'First insert Teechnology';
    END IF;
END;
```

2. **Insert new subscription data.**

```
CREATE OR REPLACE PROCEDURE p_insert_subscription (
    v_tech_id     IN NUMBER,
    sub_type      IN VARCHAR2,
    is_free       IN CHAR,
    sub_name      IN VARCHAR2,
    download_url  IN VARCHAR2,
    prerequisites IN VARCHAR2,
    p_message     OUT VARCHAR2
) IS
    v_cnt NUMBER(2);
BEGIN
    SELECT
       COUNT(1)
    INTO v_cnt
    FROM
       technology
    WHERE
       tech_id = v_tech_id;

    IF v_cnt > 0 THEN
       INSERT INTO subscription (
          subscription_id,
          tech_id,
          sub_type,
          is_free,
          sub_name,
          download_url,
          prerequisites
       ) VALUES (
          seq_subscription_id.NEXTVAL,
          v_tech_id,
          sub_type,
          is_free,
          sub_name,
          download_url,
          prerequisites
       );

       COMMIT;
       p_message := 'Added Successfully!!!';
    ELSE
       p_message := 'First insert Technology';
    END IF;
END;
```

3. **Insert new certification data.**

```
CREATE OR REPLACE PROCEDURE p_insert_certifciation (
   p_tech_id    IN NUMBER,
   p_cert_code IN VARCHAR2,
   p_cert_name    IN VARCHAR2,
   p_cert_seq_in_path   IN   NUMBER,
   p_registration_url    IN VARCHAR2,
   p_price   IN NUMBER,
   p_currency    IN VARCHAR2,
   p_exam_mode    IN VARCHAR2,
   p_exam_duration  IN  NUMBER,
   p_duration_unit  IN  VARCHAR2,
   p_message    OUT VARCHAR2
) IS
   v_cnt     NUMBER(2);
BEGIN
   SELECT
       COUNT(1)
   INTO v_cnt
   FROM
       certification
   WHERE
       tech_id = p_tech_id;

   IF v_cnt > 0 THEN

       INSERT INTO learning (
           cert_id,
           tech_id,
           cert_code,
           cert_name,
           cert_path_id,
           cert_seq_in_path,
           registration_url,
           price,
           currency,
           exam_mode,
           exam_duration,
           duration_unit
       ) VALUES (
           seq_cert_id.NEXTVAL,
           p_tech_id,
           p_cert_code,
           p_cert_name,
           SEQ_SCERT_PATH_ID.NEXTVAL,
           p_cert_seq_in_path,
           p_registration_url,
           p_price,
           p_currency,
           p_exam_mode,
           p_exam_duration,
           p_duration_unit
       );
       COMMIT;
       p_message := 'Added Successfully!!!';
   ELSE
       p_message := 'First insert Teechnology';
   END IF;

END;
```

4. **Update the price of certification data.**

```
CREATE OR REPLACE PROCEDURE p_update_cert_price (
    p_learn_id IN NUMBER,
    p_price    IN NUMBER,
    p_message  OUT VARCHAR2
) IS
    v_cnt NUMBER(2);
BEGIN
    SELECT
        COUNT(1)
    INTO v_cnt
    FROM
        learning
    WHERE
        learn_id = p_learn_id;

    IF v_cnt > 0 THEN
        IF p_price IS NULL OR p_price = 0 THEN
            UPDATE learning
            SET
                is_free = 'Y',
                price = p_price
            WHERE
                learn_id = p_learn_id;

            COMMIT;
        ELSE
            UPDATE learning
            SET
                price = p_price
            WHERE
                learn_id = p_learn_id;

            COMMIT;
            p_message := 'Updated Successfully!!!';
        END IF;
```

```
   ELSE
      p_message := 'Provide correct certificate ID';
   END IF;
END;
```

5. **Update the address data of the customer. When the argument is S, it updates street name, when the argument is P it updates phone number, when the argument is B it updates building name and when the argument is A it updates apartment number.**

```
CREATE OR REPLACE PROCEDURE p_update_local_address (
   p_cust_id IN NUMBER,
   p_string  IN VARCHAR2,
   p_field   IN CHAR, --S for street name, P for phone, B for Building name, A for
Apartmanet Number
   p_message OUT VARCHAR2
) IS
   v_cnt NUMBER(2);
BEGIN
   SELECT
      COUNT(1)
   INTO v_cnt
   FROM
      customers
   WHERE
      cust_id = p_cust_id;

   IF v_cnt > 0 THEN
      IF p_field = 'P' THEN
         UPDATE local_address
         SET
            phone = to_number(p_string)
         WHERE
            cust_id = p_cust_id;

         p_message := 'Updated Successfully!!!';
         COMMIT;
      ELSIF p_field = 'A' THEN
         UPDATE local_address
         SET
            apartment_num = to_number(p_string)
         WHERE
            cust_id = p_cust_id;

         p_message := 'Updated Successfully!!!';
         COMMIT;
      ELSIF p_field = 'B' THEN
         UPDATE local_address
         SET
            building_name = p_string
         WHERE
```

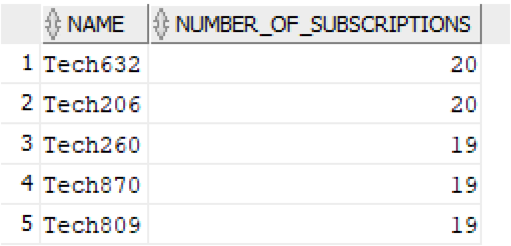
```
                cust_id = p_cust_id;

            p_message := 'Updated Successfully!!!';
            COMMIT;
        ELSIF p_field = 'S' THEN
            UPDATE local_address
            SET
                street_name = p_string
            WHERE
                cust_id = p_cust_id;

            COMMIT;
            p_message := 'Updated Successfully!!!';
        END IF;

    ELSE
        p_message := 'Provide correct customer ID';
    END IF;

END;
```

PERFORMANCE TUNING

INDEXING

1. Let's write a query to fetch top 5 technologies having maximum subscription options.

```
SELECT
   t.name,
   COUNT(1) number_of_subscriptions
FROM
        db888.subscription s
   JOIN db888.technology t ON ( s.tech_id = t.tech_id )
GROUP BY
   t.name
ORDER BY
   COUNT(1) DESC
FETCH FIRST 5 ROWS ONLY;
```

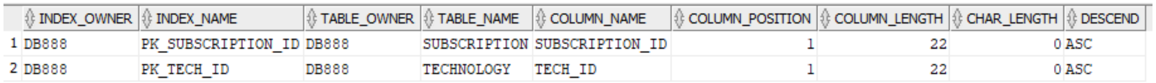


	NAME	NUMBER_OF_SUBSCRIPTIONS
1	Tech632	20
2	Tech206	20
3	Tech260	19
4	Tech870	19
5	Tech809	19

Figure 21: Fetch top 5 technologies having maximum subscription options

2. Checking the index on SUBSCRIPTION and TECHNOLOGY tables. We can observe SUBSCRIPTION table has no index on column TECH_ID which is the part of SQL above.

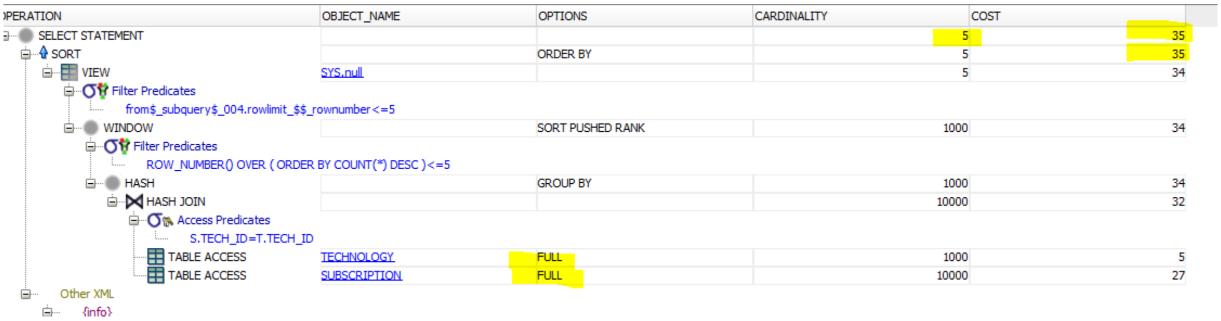
```
SELECT
   *
FROM
   all_ind_columns
WHERE
   table_name IN ( 'SUBSCRIPTION', 'TECHNOLOGY' )
   AND index_owner = 'DB888';
```



	INDEX_OWNER	INDEX_NAME	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION	COLUMN_LENGTH	CHAR_LENGTH	DESCEND
1	DB888	PK_SUBSCRIPTION_ID	DB888	SUBSCRIPTION	SUBSCRIPTION_ID	1	22	0	ASC
2	DB888	PK_TECH_ID	DB888	TECHNOLOGY	TECH_ID	1	22	0	ASC

Figure 22: Checking the index on SUBSCRIPTION and TECHNOLOGY tables

3. Performance of the query without the index on the TECH_ID column.



PERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	35
SORT		ORDER BY	5	35
VIEW	SYS.null		5	34
Filter Predicates				
from$_subquery$_004.rowlimit_$$_rownumber<=5				
WINDOW		SORT PUSHED RANK	1000	34
Filter Predicates				
ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC)<=5				
HASH		GROUP BY	1000	34
HASH JOIN			10000	32
Access Predicates				
S.TECH_ID=T.TECH_ID				
TABLE ACCESS	TECHNOLOGY	FULL	1000	5
TABLE ACCESS	SUBSCRIPTION	FULL	10000	27
Other XML				
{info}				

Figure 23: Performance of the query without the index on TECH_ID column

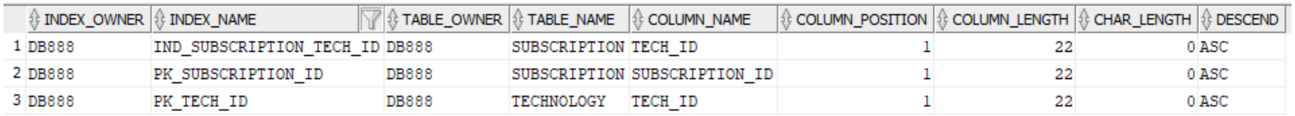
4. Let’s create an index on the TECH_ID column of table SUBSCRIPTION.

```
CREATE
INDEX ind_subscription_tech_id
ON
   db888.subscription (
      tech_id);
```

5. Checking the index again on subscription.

From the query result, we can see a record of the index-

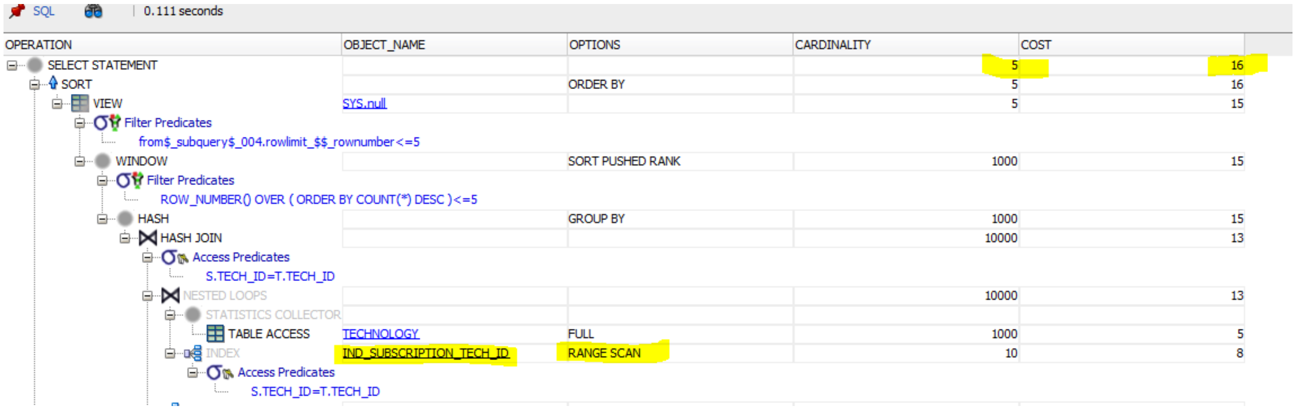
```
SELECT
   *
FROM
   all_ind_columns
WHERE
   table_name IN ( 'SUBSCRIPTION', 'TECHNOLOGY' )
   AND index_owner = 'DB888'
```



	INDEX_OWNER	INDEX_NAME	TABLE_OWNER	TABLE_NAME	COLUMN_NAME	COLUMN_POSITION	COLUMN_LENGTH	CHAR_LENGTH	DESCEND
1	DB888	IND_SUBSCRIPTION_TECH_ID	DB888	SUBSCRIPTION	TECH_ID	1	22	0	ASC
2	DB888	PK_SUBSCRIPTION_ID	DB888	SUBSCRIPTION	SUBSCRIPTION_ID	1	22	0	ASC
3	DB888	PK_TECH_ID	DB888	TECHNOLOGY	TECH_ID	1	22	0	ASC

Figure 24: Checking index on Subscription

6. Performance of the query after creating the index.



SQL | 0.111 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			5	16
SORT		ORDER BY	5	16
VIEW	SYS.null		5	15
Filter Predicates				
from$_subquery$_004.rowlimit_$$_rownumber<=5				
WINDOW		SORT PUSHED RANK	1000	15
Filter Predicates				
ROW_NUMBER() OVER (ORDER BY COUNT(*) DESC)<=5				
HASH		GROUP BY	1000	15
HASH JOIN			10000	13
Access Predicates				
S.TECH_ID=T.TECH_ID				
NESTED LOOPS			10000	13
STATISTICS COLLECTOR				
TABLE ACCESS	TECHNOLOGY	FULL	1000	5
INDEX	IND_SUBSCRIPTION_TECH_ID	RANGE SCAN	10	8
Access Predicates				
S.TECH_ID=T.TECH_ID				

Figure 25: Performance tuning of the query after creating the index

From the above results, we can see that Indexing helps in optimizing query execution time as the **cost decreased from 35 to 16 which is more than a 50% reduction. This will be much more significant when we have huge data.**

7. Looking at the requirements and our views, functions, and procedures we have created the below-listed indexes for performance improvement.

```
CREATE INDEX ind_certification_tech_id ON
    db888.certification (
        tech_id
    );

CREATE INDEX ind_local_address_cust_id ON
    local_address (
        cust_id
    );

CREATE INDEX ind_customers_first_name ON
    customers (
        first_name
    );

CREATE INDEX ind_customers_last_name ON
    customers (
        last_name
    );
```

TABLE PARTITION

The table is partitioned to keep a set of data in a specific partition. So, when we try to query data that is in the partition, the execution plan reads only the data of that partition instead of the full table. This is useful when we have huge data.

We can see the technology table is referenced by many tables and it has the column LAUNCH_DATE which has date data. We can create partitions on this column to store the data in a range of dates.

1. **First take back up of TECHNOLOGY table data.**

```
CREATE TABLE technology_bkp
   AS
      SELECT
         *
      FROM
         technology;
```

2. **Let's drop all the foreign key constraints that refer to TECH_ID of this TECHNOLOGY table.**

```
ALTER TABLE tech_stack_map DROP CONSTRAINT fk_stack_map_tech_id;

ALTER TABLE use_cases DROP CONSTRAINT fk_use_tech_id;

ALTER TABLE learning DROP CONSTRAINT fk_learn_tech_id;

ALTER TABLE subscription DROP CONSTRAINT fk_sub_tech_id;

ALTER TABLE certification DROP CONSTRAINT fk_cert_tech_id;

ALTER TABLE technology_type DROP CONSTRAINT fk_type_tech_id;

ALTER TABLE owner_companies DROP CONSTRAINT fk_own_comp_tech_id;

ALTER TABLE tech_comp_map DROP CONSTRAINT fk_comp_map_tech_id;
```

3. **Drop table TECHNOLOGY.**

```
DROP TABLE db888.technology;
```

4. **Recreate table TECHNOLOGY with partitions.**

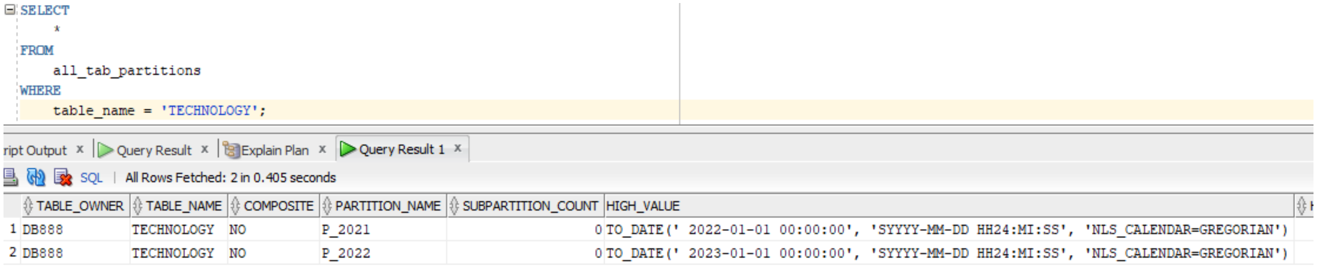
```
CREATE TABLE technology (
   tech_id        NUMBER(10),
   name           VARCHAR2(50),
   launch_date    DATE,
   is_open_source CHAR(1),
   download_site  VARCHAR2(4000) NOT NULL
)
   PARTITION BY RANGE (
      launch_date
   )
   ( PARTITION p_2021
      VALUES LESS THAN ( TO_DATE('01-01-2022', 'DD-MM-YYYY') ),
   PARTITION p_2022
      VALUES LESS THAN ( TO_DATE('01-01-2023', 'DD-MM-YYYY') )
   );
```

5. **Insert data from the backup table to the TECHNOLOGY table.**

```
INSERT INTO db888.technology (
          tech_id,
          name,
          launch_date,
          is_open_source,
          download_site
          )
SELECT
          tech_id,
          name,
          launch_date,
          is_open_source,
          download_site
FROM
          db888.technology_bkp;
```

6. **Let’s check the partitions.**

```
SELECT
    *
FROM
    all_tab_partitions
WHERE
    table_name = 'TECHNOLOGY';
```



```
SELECT
    *
FROM
    all_tab_partitions
WHERE
    table_name = 'TECHNOLOGY';
```

All Rows Fetched: 2 in 0.405 seconds

	TABLE_OWNER	TABLE_NAME	COMPOSITE	PARTITION_NAME	SUBPARTITION_COUNT	HIGH_VALUE
1	DB888	TECHNOLOGY	NO	P_2021	0	TO_DATE(' 2022-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
2	DB888	TECHNOLOGY	NO	P_2022	0	TO_DATE(' 2023-01-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')

Figure 26: Output after partitioning

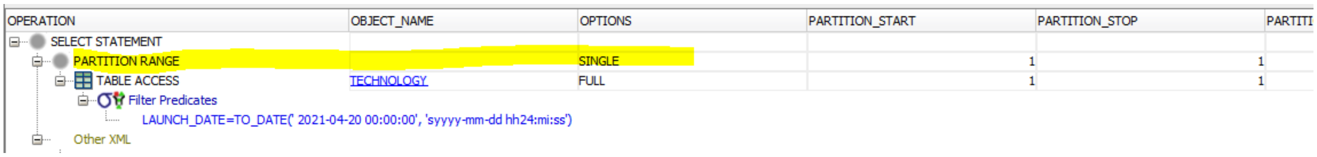
7. **Let’s recreate the constraints.**

```
ALTER TABLE technology ADD CONSTRAINT pk_tech_id PRIMARY KEY ( tech_id );
ALTER TABLE tech_stack_map
    ADD CONSTRAINT fk_stack_map_tech_id FOREIGN KEY ( tech_id )
        REFERENCES technology ( tech_id );

ALTER TABLE use_cases
    ADD CONSTRAINT fk_use_tech_id FOREIGN KEY ( tech_id )
        REFERENCES technology ( tech_id );

ALTER TABLE learning
    ADD CONSTRAINT fk_learn_tech_id FOREIGN KEY ( tech_id )
        REFERENCES technology ( tech_id );

ALTER TABLE subscription
    ADD CONSTRAINT fk_sub_tech_id FOREIGN KEY ( tech_id )
        REFERENCES technology ( tech_id );

ALTER TABLE certification
    ADD CONSTRAINT fk_cert_tech_id FOREIGN KEY ( tech_id )
        REFERENCES technology ( tech_id );

ALTER TABLE technology_type
    ADD CONSTRAINT fk_type_tech_id FOREIGN KEY ( tech_id )
        REFERENCES technology ( tech_id );

ALTER TABLE owner_companies
    ADD CONSTRAINT fk_own_comp_tech_id FOREIGN KEY ( tech_id )
        REFERENCES technology ( tech_id );
```

```
ALTER TABLE tech_comp_map
    ADD CONSTRAINT fk_comp_map_tech_id FOREIGN KEY ( tech_id )
        REFERENCES technology ( tech_id );
```

8. **Whenever we try to query the TECHNOLOGY table on column launch_date, the specific partition will be used for improved performance.**

```
SELECT
    *
FROM
    db888.technology
WHERE
    launch_date = TO_DATE('20-04-2021', 'dd-mm-yyyy');
```



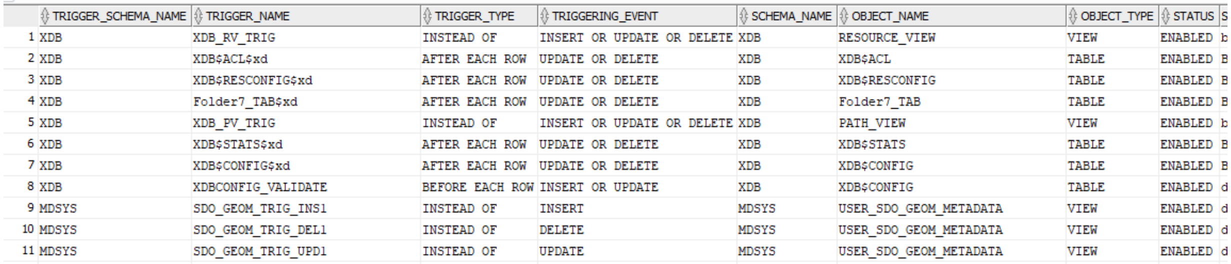
```
OPERATION                   OBJECT_NAME   OPTIONS   PARTITION_START   PARTITION_STOP   PARTITI
SELECT STATEMENT
  PARTITION RANGE                         SINGLE                  1                1
    TABLE ACCESS            TECHNOLOGY    FULL                    1                1
      Filter Predicates
        LAUNCH_DATE=TO_DATE(' 2021-04-20 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
  Other XML
```

Figure 27: Specific portioning output

DBA SCRIPTS

1. **File Name: dba/monitoring/table_triggers.sql**
   Description: Lists the triggers for the specified table.
   Call Syntax: @table_triggers (schema) (table_name)

```
SELECT
   owner           AS trigger_schema_name,
   trigger_name,
   trigger_type,
   triggering_event,
   table_owner      AS schema_name,
   table_name       AS object_name,
   base_object_type AS object_type,
   status,
   trigger_body     AS script
FROM
   sys.all_triggers;
```



	TRIGGER_SCHEMA_NAME	TRIGGER_NAME	TRIGGER_TYPE	TRIGGERING_EVENT	SCHEMA_NAME	OBJECT_NAME	OBJECT_TYPE	STATUS	S
1	XDB	XDB_RV_TRIG	INSTEAD OF	INSERT OR UPDATE OR DELETE	XDB	RESOURCE_VIEW	VIEW	ENABLED	b
2	XDB	XDB$ACL$xd	AFTER EACH ROW	UPDATE OR DELETE	XDB	XDB$ACL	TABLE	ENABLED	B
3	XDB	XDB$RESCONFIG$xd	AFTER EACH ROW	UPDATE OR DELETE	XDB	XDB$RESCONFIG	TABLE	ENABLED	B
4	XDB	Folder7_TAB$xd	AFTER EACH ROW	UPDATE OR DELETE	XDB	Folder7_TAB	TABLE	ENABLED	B
5	XDB	XDB_PV_TRIG	INSTEAD OF	INSERT OR UPDATE OR DELETE	XDB	PATH_VIEW	VIEW	ENABLED	b
6	XDB	XDB$STATS$xd	AFTER EACH ROW	UPDATE OR DELETE	XDB	XDB$STATS	TABLE	ENABLED	B
7	XDB	XDB$CONFIG$xd	AFTER EACH ROW	UPDATE OR DELETE	XDB	XDB$CONFIG	TABLE	ENABLED	B
8	XDB	XDBCONFIG_VALIDATE	BEFORE EACH ROW	INSERT OR UPDATE	XDB	XDB$CONFIG	TABLE	ENABLED	d
9	MDSYS	SDO_GEOM_TRIG_INS1	INSTEAD OF	INSERT	MDSYS	USER_SDO_GEOM_METADATA	VIEW	ENABLED	d
10	MDSYS	SDO_GEOM_TRIG_DEL1	INSTEAD OF	DELETE	MDSYS	USER_SDO_GEOM_METADATA	VIEW	ENABLED	d
11	MDSYS	SDO_GEOM_TRIG_UPD1	INSTEAD OF	UPDATE	MDSYS	USER_SDO_GEOM_METADATA	VIEW	ENABLED	d

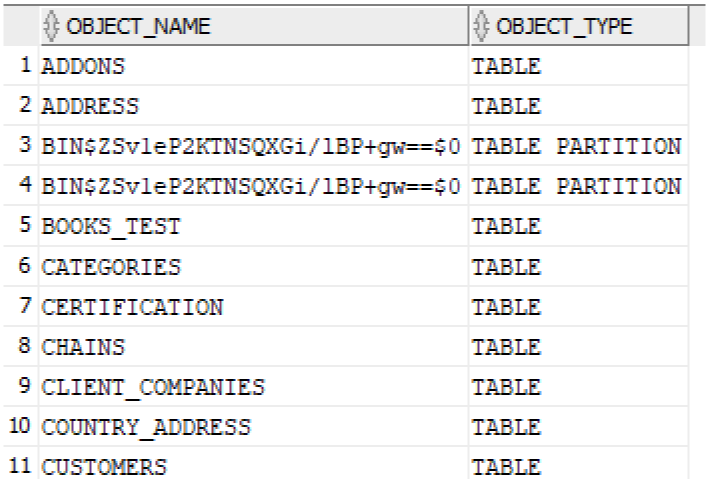
Figure 28: List of Triggers

2. **File Name: monitoring/user_objects.sql**
   Description: Displays the objects owned by the current user.

```
SELECT
   object_name,
   object_type
FROM
   user_objects
ORDER BY
   1,
   2;
```

Description: The above SQL accesses user_objects and displays object names and object types for the current user.

Result:



	OBJECT_NAME	OBJECT_TYPE
1	ADDONS	TABLE
2	ADDRESS	TABLE
3	BIN$ZSvleP2KTNSQXGi/1BP+gw==$0	TABLE PARTITION
4	BIN$ZSvleP2KTNSQXGi/1BP+gw==$0	TABLE PARTITION
5	BOOKS_TEST	TABLE
6	CATEGORIES	TABLE
7	CERTIFICATION	TABLE
8	CHAINS	TABLE
9	CLIENT_COMPANIES	TABLE
10	COUNTRY_ADDRESS	TABLE
11	CUSTOMERS	TABLE

Figure 29:List of Objects owned by the current user

3. **File Name: dba/monitoring/system_privs.sql**
   Description: Displays users granted the specified system privilege.
   Requirements: Access to the DBA views.
   Call Syntax: @system_privs ("sys-priv")

```
SELECT
   privilege,
   grantee,
   admin_option
FROM
   dba_sys_privs
WHERE
   privilege LIKE upper('%&1%')
ORDER BY
   privilege,
   grantee;
```

Description: The above sql which access dba_sys_privs and displays the users granted the specified system privilege.

4. **File Name: monitoring/table_stale_tables.sql**
   Description: Displays stale partition names which need to be analyzed for performance.

```
SELECT
   owner,
   table_name,
   partition_name,
   subpartition_name,
   stale_stats
FROM
   all_tab_statistics
WHERE
   stale_stats = 'YES';
```

Description: The above sql which access all_tab_statistics and displays table names along with partition and sub-partition names that are in STALE status and needs to be analyzed for better execution plan and hence performance.

Thank You